Document Title	Specification of Crypto Service Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	402
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Change Description
2018-10-31	4.4.0	AUTOSAR Release Management	• Client-Server-Interfaces Csm<Service>_{Config} • corrected CS interfaces • removal of references to CryptoAbstractionLibrary
2017-12-08	4.3.1	AUTOSAR Release Management	• Added definition for asymmetric key formats • Error fixing and consistency improvements • Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Introduced crypto job concept • Introduced key management concept • Removed Cry_XXX functions from the Csm and introduced two new layers in the crypto stack: Crypto Interface (CryIf) and Crypto Driver (Crypto)
2015-07-31	4.2.2	AUTOSAR Release Management	• Changed return type from Csm_ReturnType to Std_Types in all API functions • Added detailed description of RTE interfaces • Debugging support marked as obsolete • Error fixing and consistency improvements

Document Change History

Date	Release	Changed by	Change Description
2014-10-31	4.2.1	AUTOSAR Release Management	• Obsolete configuration elements removed • Error fixing and consistency improvements • Editorial changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Error fixing and consistency improvements • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Error fixing and consistency improvements • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Services for compression/decompression added • Services for key update added (Concept 'CSM extension') • Services for symmetric key generation added (Concept 'CSM extension') • Service state machine changed to cope with terminated users by releasing of locked resources • Production errors restructured
2011-12-22	4.0.3	AUTOSAR Administration	• Fixed issues with AUTOSAR Port Interfaces
2010-09-30	3.1.5	AUTOSAR Administration	• Complete Configuration parameters • Complete API specifications • Add support for secure key storage • Integration of support for key transport services • Introduction of new DET error (checking of the null pointer in getversion info).
2010-02-02	3.1.4	AUTOSAR Administration	• Initial release

Disclaimer

Table of Contents

1 Introduction and Functional Overview

This specification specifies the functionality, API and the configuration of the software module Crypto Service Manager (CSM) to satisfy the top-level requirements represented in the CSM Requirements Specification (SRS) [CSM_SRS].

The CSM shall provide synchronous or asynchronous services to enable a unique access to basic cryptographic functionalities for all software modules. The CSM shall provide an abstraction layer, which offers a standardized interface to higher software layers to access these functionalities.

The functionality required by a software module can be different to the functionality required by other software modules. For this reason, there shall be the possibility to configure and initialize the services provided by the CSM individually for each software module. This configuration comprises as well the selection of synchronous or asynchronous processing of the CSM services.

The construction of the CSM module follows a generic approach. Wherever a detailed specification of structures and interfaces would limit the scope of the usability of the CSM, interfaces and structures are defined in a generic way. This provides an opportunity for future extensions.

2 Acronyms and Abbreviations

Acronyms and abbreviations, which have a local scope and therefore are not contained in the AUTOSAR glossary [13], are listed in this chapter.

Abbreviation / Acronym:	***Description:***
AEAD	Authenticated Encryption with Associated Data
CDD	Complex Device Driver
CSM	Crypto Service Manager
CRYIF	Crypto Interface
CRYPTO	Crypto Driver
DET	Default Error Tracer
HSM	Hardware Security Module
HW	Hardware
SHE	Security Hardware Extension
SW	Software

2.1 Glossary of Terms

Terms:	***Description:***	
Crypto Driver Object	A Crypto Driver implements one or more Crypto Driver Objects. The Crypto Driver Object can offer different crypto primitives in hardware or software. The Crypto Driver Objects of one Crypto Driver are independent of each other. There is only one workspace for each Crypto Driver Object (i.e. only one crypto primitive can be performed at the same time)	
Key	A Key can be referenced by a job in the Csm. In the Crypto Driver, the key refers a specific key type.	
Key Type	A key type consists of refers to key elements. The key types are typically pre-configured by the vendor of the Crypto Driver.	
Key Element	Key elements are used to store data. This data can be e.g. key material or the IV needed for AES encryption. It can also be used to configure the behaviour oft he key management functions.	
Job	A job is a configured Object with refers to a key and a cryptographic primitive.	
Channel	A channel is the path from a Crypto Service Manager queue via the Crypto Interface to a specific Crypto Driver Object.	
Crypto Primitive	A crypto primitive is an instance of a configured cryptographic algorithm realized in a Crypto Driver Object.	
Operation	An operation of a crypto primitive declares what part of the crypto primitive shall be performed. There are three different operations:	
	START	Operation indicates a new request of a crypto primitive, it shall cancel all previous requests perform necessary

		initializations and checks if the crypto primitive can be processed.
	UPDATE	Operation indicates, that the crypto primitive expect input data. An update operation may provide intermediate results.
	FINISH	Operation indicates, that after this part all data are fed completely and the crypto primitive can finalize the calculations. A finish operation may provide final results.
	It is also possible to perform more than one operation at once by concatenating the corresponding bits of the operation_mode argument.	
Priority	The priority of a job defines the importance of it. The higher the priority (as well in value), the more immediate the job will be executed. The priority of a cryptographic job is part of the configuration.	
Processing	Indicates the kind of job processing.	
	Asynchronous	The job is not processed immediately when calling a corresponding function. Usually, the caller is informed via a callback function when the job has been finished.
	Synchronous	The job is processed immediately when calling a corresponding function. When the function returns, a result will be available.

3 Related documentation

3.1 Input Documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Specification of RTE Software
AUTOSAR_SWS_RTE.pdf

[5] Specification of BSW Scheduler
AUTOSAR_SWS_Scheduler.pdf

[6] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[7] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[8] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.doc.pdf

[9] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf

[10] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[11] Specification of C Implementation Rules
AUTOSAR_TR_CImplementationRules.pdf

[12] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[13] AUTOSAR Glossary
AUTOSAR_TR_Glossary.pdf

[14] Requirements on the Crypto Stack
AUTOSAR_SRS_CryptoStack.pdf

[15] Specification of the Crypto Interface
AUTOSAR_SWS_CryptoInterface.pdf

[16] Specification of the Crypto Driver
AUTOSAR_SWS_CryptoDriver.pdf

[17] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

[18] IEC 7498-1 The Basic Model, IEC Norm, 1994

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules (SWS BSW General), which is also valid for Crypto Service Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Crypto Service Manager.

4 Constraints and Assumptions

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.

4.2 Applicability to Car Domains

n.a.

4.3 Security Implications

There is no user management in place, which prevents non-authorized access on any of CSM’s services. This means, that if any access protection is needed such must be implemented by the application and the served (by CSM) cryptographic library modules; access protection is not target of the CSM.

5 Dependencies to other Modules

[SWS_Csm_00001] ⌈The CSM shall be able to access the cryptographic interface (CRYIF), which is implemented according to the cryptographic interface specification. ⌋(SRS_CryptoStack_00082)

[SWS_Csm_00506] ⌈The CSM module shall use the interfaces of the CRYIF with the underlying Crypto Drivers (CRYPTO) to calculate the result of a cryptographic service.
⌋(SRS_CryptoStack_00082)
The incorporated cryptographic library modules or hardware extensions of the Crypto Driver provide the cryptographic routines, e.g. SHA-1, RSA, AES, Diffie-Hellman key-exchange, etc.

5.1 File Structure

5.1.1 Code File Structure

[SWS_Csm_00002] ⌈The code file structure shall not be defined within this specification completely. The CSM module shall consist of the following parts:
⌋()

6 Requirements Traceability

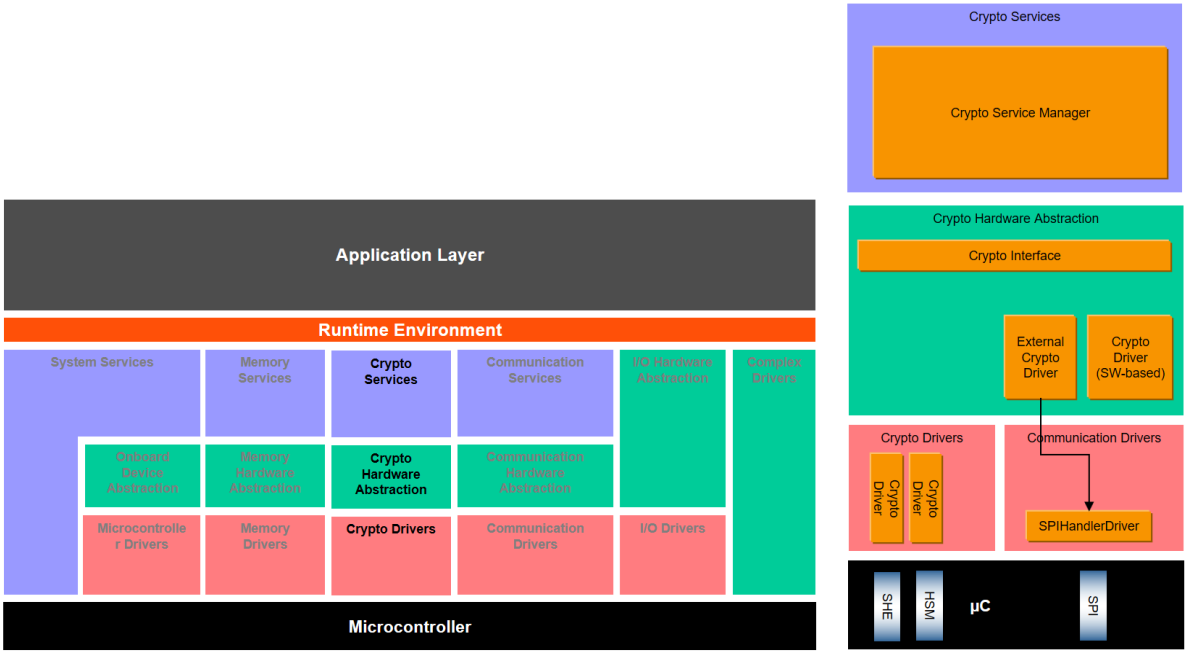
Requirement	Description	Satisfied by
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Csm_00646
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Csm_00646
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Csm_00073, SWS_Csm_00970, SWS_Csm_00971
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Csm_00073, SWS_Csm_00970, SWS_Csm_00971
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Csm_00479
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Csm_00705
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_Csm_00646
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Csm_00479
SRS_CryptoStack_00008	The Crypto Stack shall allow static configuration of keys used for cryptographic jobs	SWS_Csm_00951, SWS_Csm_00953, SWS_Csm_01012
SRS_CryptoStack_00009	The Crypto Stack shall support reentrancy for all crypto services	SWS_Csm_00022
SRS_CryptoStack_00010	The Crypto Stack shall conceal symmetric keys from the users of crypto services	SWS_Csm_00959
SRS_CryptoStack_00011	The Crypto Stack shall conceal asymmetric private keys from the users of Crypto services	SWS_Csm_00959
SRS_CryptoStack_00019	The Crypto Stack shall identify random number generation as	SWS_Csm_01543

	a cryptographic primitive which can be requested to a driver	
SRS_CryptoStack_00020	The Crypto Stack shall identify symmetric encryption/decryption as a cryptographic primitive which can be requested to a driver	SWS_Csm_00984, SWS_Csm_00989
SRS_CryptoStack_00021	The Crypto Stack shall identify asymmetric encryption/decryption as a cryptographic primitive which can be requested to a driver	SWS_Csm_00984, SWS_Csm_00989
SRS_CryptoStack_00022	The Crypto Stack shall identify MAC generation/verification as a cryptographic primitive which can be requested to a driver	SWS_Csm_00982
SRS_CryptoStack_00023	The Crypto Stack shall identify asymmetric signature generation/verification as a cryptographic primitive which can be requested to a driver	SWS_Csm_00992, SWS_Csm_00996
SRS_CryptoStack_00024	The Crypto Stack shall identify hash calculation as a cryptographic primitive which can be requested to a driver	SWS_Csm_00980
SRS_CryptoStack_00026	The Crypto Stack shall provide an interface for the generation of asymmetric keys	SWS_Csm_00955
SRS_CryptoStack_00027	The Crypto Stack shall provide an interface for the generation of symmetric keys	SWS_Csm_00955
SRS_CryptoStack_00082	The CSM module specification shall specify the interface and behavior of the callback function, if the asynchronous job processing mode is selected	SWS_Csm_00001, SWS_Csm_00032, SWS_Csm_00506
SRS_CryptoStack_00084	The CSM module shall use the streaming approach for some selected services	SWS_Csm_01039
SRS_CryptoStack_00086	The CSM module shall distinguish between error types	SWS_Csm_01089, SWS_Csm_91004
SRS_CryptoStack_00087	The CSM module shall report detected development errors to the Default Error Tracer	SWS_Csm_01088, SWS_Csm_91012
SRS_CryptoStack_00090	The CSM shall provide an interface to be accessible via the RTE	SWS_Csm_00073, SWS_Csm_00802, SWS_Csm_00803, SWS_Csm_00902, SWS_Csm_00903, SWS_Csm_00912, SWS_Csm_00922, SWS_Csm_00923, SWS_Csm_00927, SWS_Csm_00928,

		SWS_Csm_00930, SWS_Csm_00934, SWS_Csm_00935, SWS_Csm_00936, SWS_Csm_00943, SWS_Csm_00946, SWS_Csm_01042, SWS_Csm_01074, SWS_Csm_01075, SWS_Csm_01077, SWS_Csm_01078, SWS_Csm_01079, SWS_Csm_01906, SWS_Csm_01910, SWS_Csm_01915, SWS_Csm_01920, SWS_Csm_01921, SWS_Csm_01922, SWS_Csm_01923, SWS_Csm_01924, SWS_Csm_01925, SWS_Csm_01926, SWS_Csm_01927, SWS_Csm_01928, SWS_Csm_09000, SWS_Csm_91023, SWS_Csm_91051, SWS_Csm_91052, SWS_Csm_91053, SWS_Csm_91054, SWS_Csm_91055, SWS_Csm_91056, SWS_Csm_91057, SWS_Csm_91058, SWS_Csm_91059, SWS_Csm_91060, SWS_Csm_91061, SWS_Csm_91062
SRS_CryptoStack_00091	The CSM shall provide one Provide--Port for each configuration	SWS_Csm_00934, SWS_Csm_01042, SWS_Csm_91023, SWS_Csm_91062
SRS_CryptoStack_00095	The Crypto Driver module shall strictly separate error and status information	SWS_Csm_01069, SWS_Csm_91001
SRS_CryptoStack_00100	Synchronous Job Processing	SWS_Csm_01049
SRS_CryptoStack_00101	Asynchronous Job Processing	SWS_Csm_01049
SRS_CryptoStack_00102	The priority of a user and its crypto jobs shall be defined by static configuration	SWS_Csm_01010
SRS_CryptoStack_00103	The Crypto Stack shall provide an interface for the derivation of symmetric keys	SWS_Csm_00956
SRS_CryptoStack_00906	-	SWS_Csm_00947
SRS_CryptoStack_01076	-	SWS_Csm_01083
SRS_CrytptoStack_00028	-	SWS_Csm_00966, SWS_Csm_00967
SRS_CrytptoStack_00029	-	SWS_Csm_00959
SRS_CrytptoStack_00031	-	SWS_Csm_01036
SRS_Csm_00066	-	SWS_Csm_00691, SWS_Csm_00728, SWS_Csm_01905
SWS_BSW_00050	Check parameters passed to Initialization functions	SWS_Csm_00186
SWS_BSW_00216	-	SWS_Csm_01085

7 Functional specification

AUTOSAR Layered View [2].



AUTOSAR Layered View with CSM

7.1 Basic Architecture Guidelines

The starting point for the description of the design of the CSM module is the AUTOSAR Layered Software Architecture (see Figure AUTOSAR Layered View). The description of the CSM module architecture on the basis of the AUTOSAR layered software architecture shall help to understand the specification of interfaces and functionalities of the CSM module in the following sections.

The architecture of AUTOSAR consists of several layers which can be seen in Figure AUTOSAR Layered View. The Service Layer is the highest layer of the Basic Software. Its task is to provide basic services for application and basic software modules, i.e. it offers the most relevant functionalities for application software and basic software modules.

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.

7.2 General Behavior

[SWS_Csm_00941] [A job is an instance of a configurated cryptographic primitive.

](

[SWS_Csm_00016] [For each job just one instance shall be processed by CSM at a time.
](

[SWS_Csm_00022] [The CSM module shall allow parallel processing of different jobs.
](SRS_CryptoStack_00009)

[SWS_Csm_00017] [If a service of the CSM module is requested and the corresponding job is being processed, the job request shall be rejected with the return value `CRYPTO_E_BUSY`.
](

Note: “job is being processed” means that the corresponding crypto driver object is currently and actively processing this job. When a job is not finished but the crypto driver object is not active with it (because, e.g., the operation “FINISH” is outstanding), this does not mean that this job is being processed.

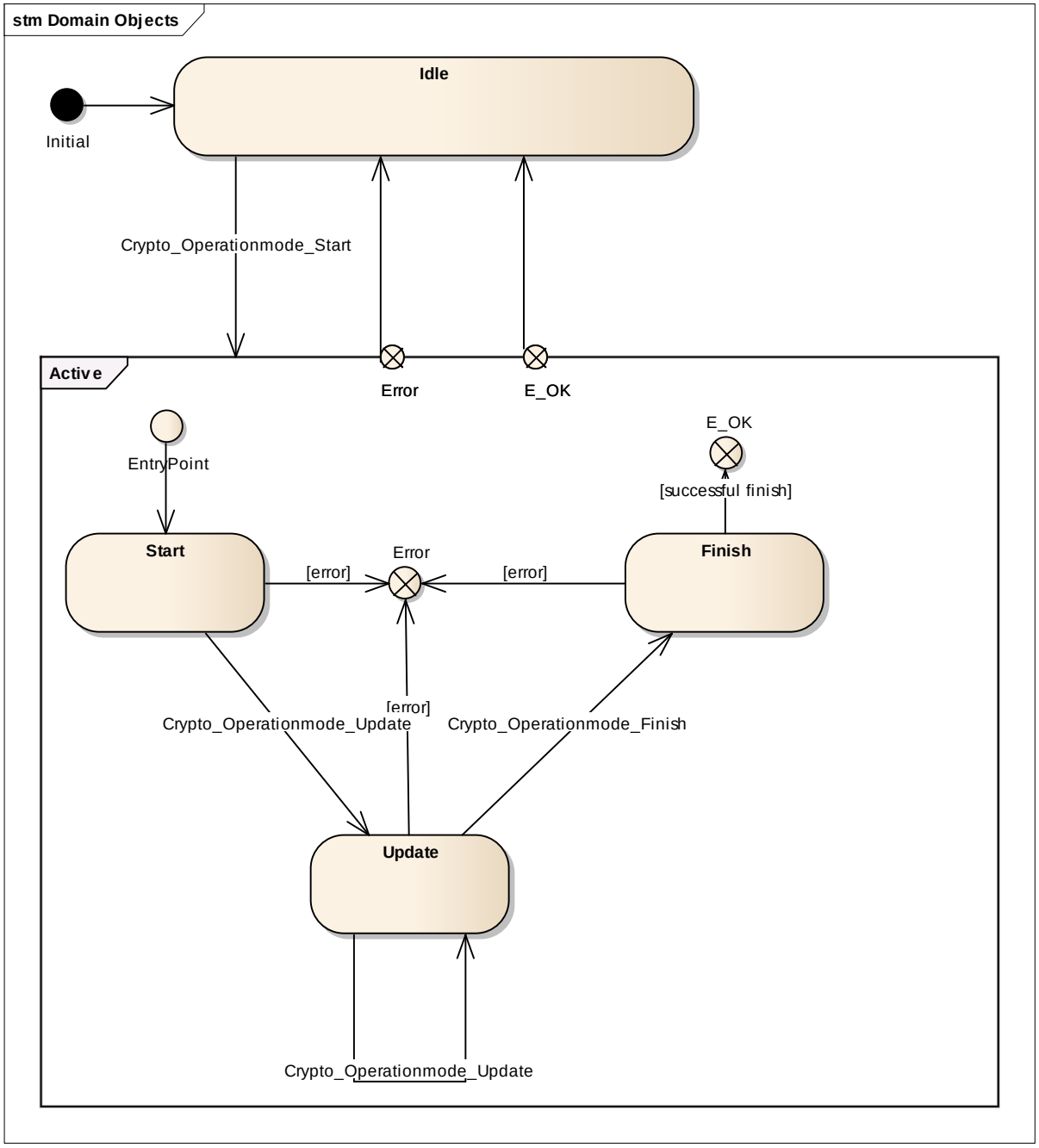
[SWS_Csm_00019] [If an asynchronous interface is configured, the CSM module shall provide a main function Csm_MainFunction() which is called cyclically to control processing of the jobs via a state machine.
](

7.2.1 Normal Operation

[SWS_Csm_01039] [To unite a single call function and the streaming approach for the crypto services, there is the `mode` parameter, which determines the operation mode. This service operation is a flag field, indicating the operation mode “START”, “UPDATE” or “FINISH”. It declares explicitly what operation shall be performed. These operation modes can be mixed, and execute multiple operations at once. The diagram in **SWS_Csm_00024** shows the state machine of a job of this design.
](SRS_CryptoStack_00084)

Note: The actual transaction of the states is made in the layer, which works with these states, i.e. in the Crypto Driver.

[SWS_Csm_00024] [



](

[**SWS_Csm_01033]**[The CSM crypto services shall support to process multiple operation mode inputs with a single call.
](

[SWS_Csm_01045][If the `CRYPTO_OPERATIONMODE_START` and `CYRPTO_OPERATIONMODE_FINISH` bits are set and the `CRYPTO_OPERATIONMODE_UPDATE` is not set, the `Csm_<Service>()` function shall return with `E_NOT_OK`.
](

Note: The coherent single call approach could improve the performance due to less overhead. Instead of calling the explicit API multiple times, only one call is necessary. This approach is intended to be used with small data input, which demand fast processing.
While operating with the streaming approach (“Start”, “Update”, “Finish”) the dedicated Crypto Driver Object is waiting for further input (“Update”) until the “Finish” state has been reached. No other job could be processed on this Crypto Driver instance meanwhile.Functional Requirements

7.2.1.1 Configuration

[SWS_Csm_91005] ⌈Each crypto primitive configuration shall be realized as a constant structure of type .
⌋()

[SWS_Csm_91006] ⌈Each job primitive configuration shall be realized as a constant structure of type `Crypto_JobPrimitiveInfoType`.
⌋()

[SWS_Csm_00028] ⌈It shall be possible to create several configurations for each cryptographic primitive.
⌋()
One configuration per job per primitive is possible.

[SWS_Csm_00029] ⌈When creating a primitive configuration, it shall be possible to configure all available and allowed schemes from the underlying Crypto Driver Object.
⌋()
[SWS_Csm_00032] ⌈If the asynchronous interface is chosen, each job primitive configuration shall contain a callback function.
⌋(SRS_CryptoStack_00082)

7.2.1.2 Synchronous Job Processing

[SWS_Csm_00035] ⌈When the synchronous interface is used, the interface functions shall immediately compute the result with the help of the underlying Crypto Stack modules.
⌋()

[SWS_Csm_00037] ⌈ If a synchronous job is issued and the priority is greater than the highest priority available in the queue, the CSM shall disable processing new jobs from the queue until the next call of the main function has finished that follows after completion of the currently processed job.
⌋()

Note:
Channels may hold jobs of both asynchronous and synchronous processing type. If

so, a synchronous job might not be accepted for processing although its job's priority is higher than those of all asynchronous jobs.

Note:
As the underlying Crypto Driver can have its own queue, it can not always be ensured that the highest priority job provided by the application is processed next.

[SWS_Csm_91007] ⌈ If a synchronous job is issued and the priority is less than the highest priority available in the queue, the CSM shall return E_BUSY.
⌋()

Note:
By pausing calls to the CSM main function with e.g. critical sections during calling the synchronous jobs, it can be ensured, that synchronous jobs can be processed in a row without having to wait for asynchronous jobs in between if they have a high enough priority. Also consider disabling queueing in the Crypto Driver Object to ensure fast processing of synchronous jobs.
If the loading of asynchronous jobs from the queue shall not be paused by synchronous jobs, the priorities of the synchronous jobs have to be smaller than the asynchronous jobs.

7.2.1.3 Asynchronous Job Processing

[SWS_Csm_00036] ⌈If the asynchronous interface is used, the interface functions shall only hand over the necessary information to the underlying Crypto Stack modules.
⌋()

[SWS_Csm_00039] ⌈The users of the CSM shall be notified when a requested cryptographic service has been processed by calling the callback function from the job primitive configuration.
⌋()

7.2.2 Design Notes

The CSM provides two services: (1) the crypto services itself and (2) key management.

7.2.2.1 CSM module startup

The `Csm_Init()` request shall not be responsible to trigger the initialization of the underlying CRYIF. It is assumed, that the underlying CRYIF will be initialized by any appropriate entity (e.g. BswM).

Software components, which are using the CSM module, shall be responsible for checking global error and status information resulting from the CSM module startup.

7.2.2.2 Crypto Services

7.2.2.2.1 Usage of the CSM crypto services

[SWS_Csm_00734]⌈CSM crypto services shall provide a `Csm_<Service>()` API.
⌋()

[SWS_Csm_00924]⌈The application shall be able to call `Csm_<Service>()` with the operation mode `CRYPTO_OPERATIONMODE_START` to initialize cryptographic computations.
⌋()

[SWS_Csm_00925]⌈The application shall be able to call Csm_<Service>() with the operation mode CRYPTO_OPERATIONMODE_UPDATE arbitrary often, but at least one time, to feed the job's crypto primitive with input data.
⌋()

[SWS_Csm_01046]⌈The application shall be able to call `Csm_<Service>()` with the operation mode `CRYPTO_OPERATIONMODE_FINISH` to finalize cryptographic computations.
⌋()

[SWS_Csm_00937] ⌈The deprecated `Csm_<Service>Start()` functions shall be mapped to the `Csm_KeyElementSet()` function and the `Csm_<Service>`() functions with the operation mode “start”.
⌋()

[SWS_Csm_00938] ⌈The deprecated `Csm_<Service>Update()` functions shall be mapped to the `Csm_<Service>`() functions with the operation mode “update”.
⌋()

[SWS_Csm_00939] ⌈The deprecated `Csm_<Service>Finish()` functions shall be mapped to the `Csm_<Service>()` functions with the operation mode “finish”.
⌋()

Note:

The Csm_<Service>() will call the CryIf_ProcessJob() with a pointer to Crypto_JobType, where all the necessary information are stored to process the job.

Part of this Crypto_JobType is a Crypto_JobPrimitiveInputOutputType, where all the information about the input and output parameters depending of the service are stored. A definition of the mapping from the API parameters of Csm_<Service>() to the parameters of Crypto_JobPrimitiveInputOutputType, can be found in [SWS_Crypto_00073] of the Crypto Driver specification.

7.2.2.2.2 Queuing

The CSM may have several queues, where the jobs are lining up depending on their priority, to process multiple cryptographic requests. The path from a CSM queue via the CryIf to a Crypto Driver Object is called a *channel*. Each queue of the CSM is mapped to one channel to access the crypto primitives of the Crypto Driver Object. The size of the queue is configurable.
To optimize the hardware usage of the Crypto Driver Object, there is optionally a queue in Crypto Driver, too.

A Crypto Driver Object represents an instance of an independent crypto “device” (hardware or software, e.g. AES accelerator). There could be a channel for fast AES and CMAC calculations on an HSM for jobs with high priority, which ends on a native AES calculation service in the Crypto Driver. But it is also possible, that a Crypto Driver Object is a piece of software, e.g. for RSA calculations where users are able to encrypt, decrypt, sign or verify data.

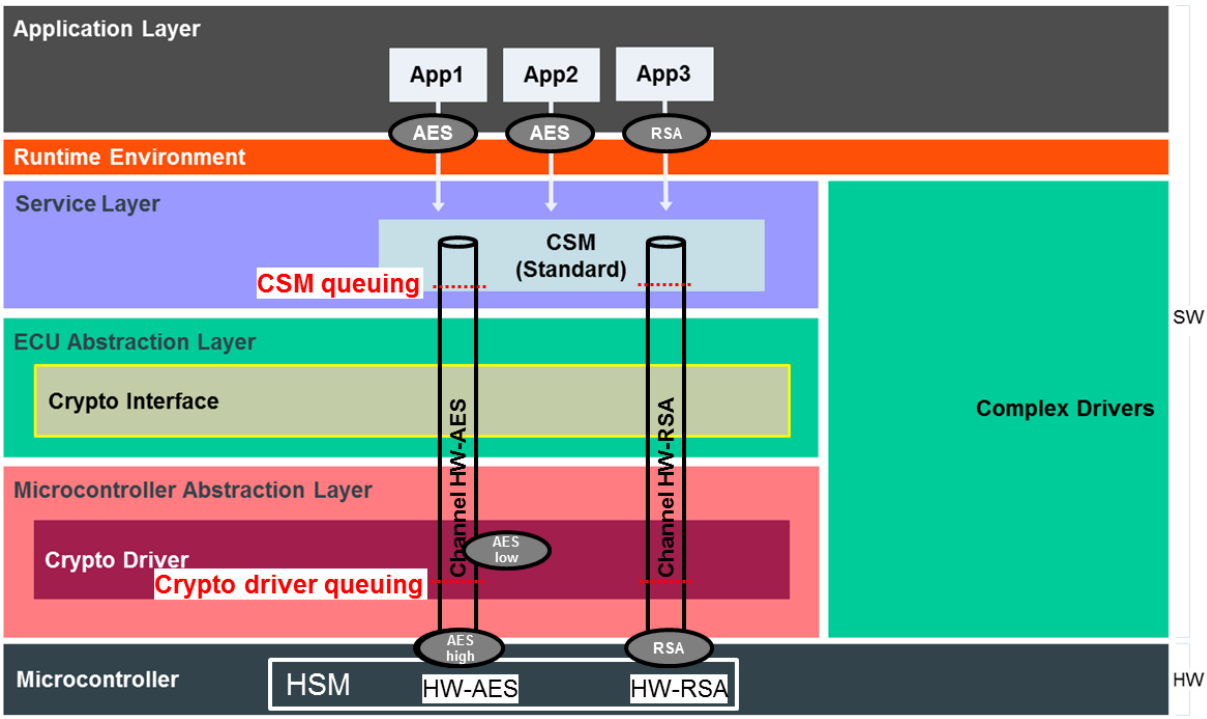


Figure 7.1 AUTOSAR Layered View with channels

Figure 7.1 illustrates an AUTOSAR Layered View with channels. In this example, there is a HSM with two Crypto Driver Objects (HW-AES and HW-RSA), each of them has an own channel. Each channel is connected to a CSM queue and a Crypto Driver Object queue.
In this case, both Crypto Driver Objects are processing a crypto job (AES-high and RSA) each, while the queue of the Crypto Driver Object contains one more job (AES-low). If the HW-AES of the HSM finished the AES-high job, AES-low job will be processed as next one.

Other scenarios with the same setup (without jobs in process or in queues) can be derived as follows:
It will be assumed, that a new job of an application calls RSA.

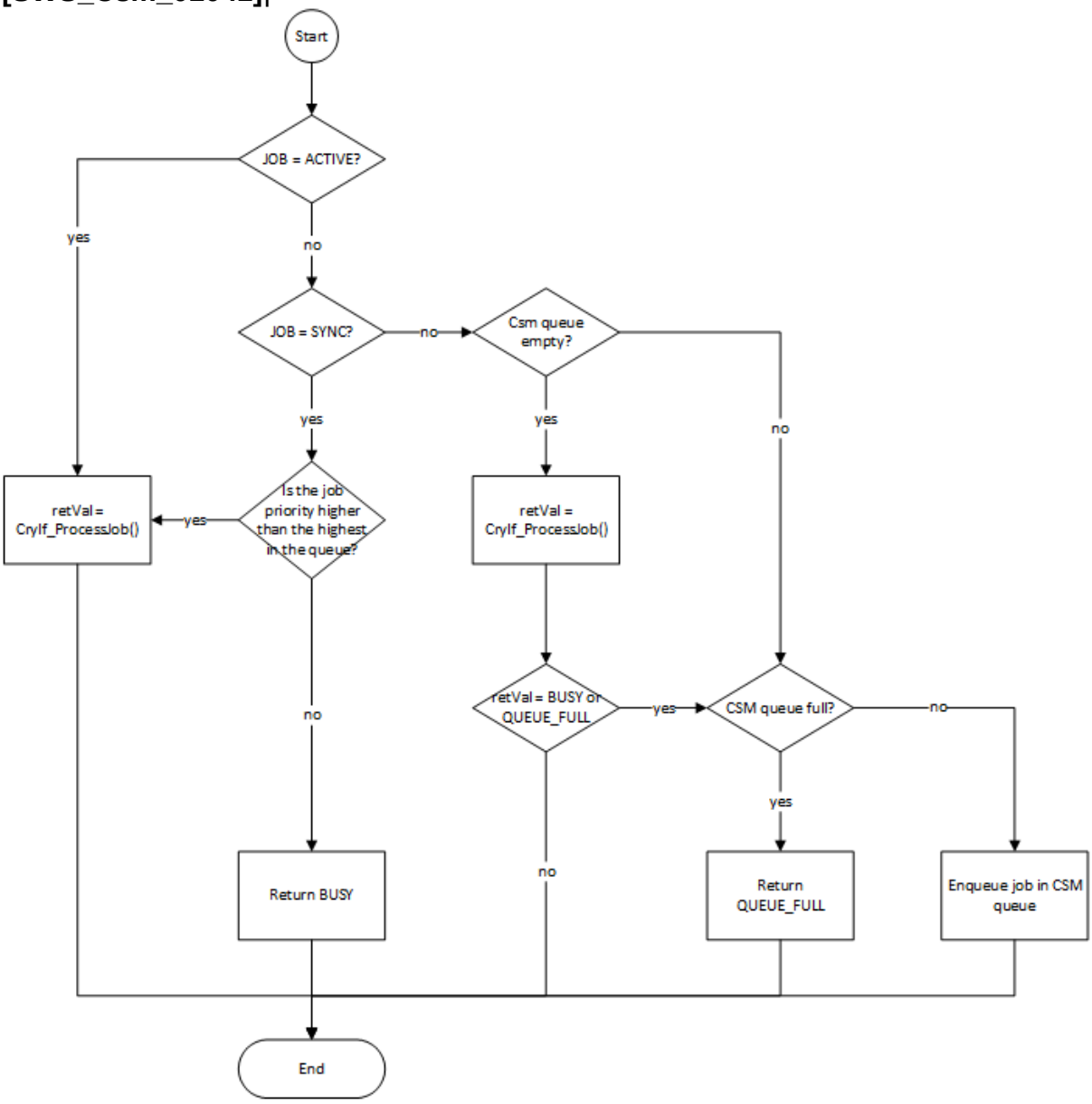
- If the Crypto Driver Object of the RSA is not busy, the job will be processed immediately.
- If the Crypto Driver Object of the RSA is busy, but the queue of the Crypto Driver Object is not full, the job will be listed into that queue in order of its priority. As soon as the Crypto Driver Object is free, the job with the highest priority from the Crypto Driver Object queue will be executed.
- If the Crypto Driver Object of the RSA is busy and the queue of the Crypto Driver Object is full, the job will be stored in the CSM queue in order of its priority.
- If the Crypto Driver Object of the RSA is busy and the queue of the Crypto Driver Object as well as the CSM queue are full, the CSM rejects the request.
- If the Crypto Driver Object of the RSA is active, the job is already started in the Crypto Driver and is waiting for either more data to process or the finish command.

[SWS_Csm_00940] ⌈It shall be possible to queue CSM jobs in configured CsmQueues in the CSM.
⌋()

[SWS_Csm_00944] ⌈The CsmQueues shall sort the jobs according to the configured job's priority.
⌋()
The higher the job priority value, the higher the job's priority.

[SWS_Csm_00945] ⌈The Csm_<Service>() function shall behave as shown in diagram SWS_Csm_01041.
⌋()

[SWS_Csm_01041][



]()

Synchronous job processing and queuing might not be useful. So, if synchronous job processing is chosen, the queue sizes should be “0”. However, it is also possible to use channels (including queues) with synchronous and asynchronous jobs.

The queued jobs can be passed to the CRYIF in the `Csm_MainFunction()`.

If the job has the state “active” the CSM shall assume, that the mapped cryptographic driver instance is currently processing this job and the caller wants to continue with the operation (e.g. feeding more data using “update”). The plausibility check has to be performed in the cryptographic driver instance.

7.2.2.3 Key Management

[SWS_Csm_00950] ⌈Services belonging to the key management shall provide the `Csm_<Service>()` function, only.
⌋()

[SWS_Csm_00954] ⌈A key consists of one or more key elements.
⌋()
Examples of key elements are the key material itself, an initialization vector, a seed for random number generation, or the proof of the SHE standard.

Keys, i.e. the corresponding key IDs have symbolic names given by the configuration. The Crypto Stack API uses the following key element index definition from the CSM module:

[SWS_Csm_01022] ⌈

Crypto Service:	*key element:*	*key element Name:*	*key element ID:*	*Mandatory:*
MAC	Key Material	CRYPTO_KE_MAC_KEY	1	x
	Proof (SHE)	CRYPTO_KE_MAC_PROOF	2	
	Seed	CRYPTO_KE_KEYGENERATE_SEED	16	
Signature	Key Material	CRYPTO_KE_SIGNATURE_KEY	1	x
Random	Seed State	CRYPTO_KE_RANDOM_SEED_STATE	3	
	Algorithm	CRYPTO_KE_RANDOM_ALGORITHM	4	
Cipher/AEAD	Key Material	CRYPTO_KE_CIPHER_KEY	1	x
	Init Vector	CRYPTO_KE_CIPHER_IV	5	
	Proof (SHE)	CRYPTO_KE_CIPHER_PROOF	6	
	2nd Key Material	CRYPTO_KE_CIPHER_2NDKEY	7	
Key Exchange	Base	CRYPTO_KE_KEYEXCHANGE_BASE	8	x
	Private Key	CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9	x
	Own Public Key	CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY	10	x
	Shared Value	CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE	1	x

	Algorithm	CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12	
Key Derivation	Password	CRYPTO_KE_KEYDERIVATION_PASSWORD	1	x
	Salt	CRYPTO_KE_KEYDERIVATION_SALT	13	
	Iterations	CRYPTO_KE_KEYDERIVATION_ITERATIONS	14	
	Algorithm	CRYPTO_KE_KEYDERIVATION_ALGORITHM	15	
Key Generate	Key Material	CRYPTO_KE_KEYGENERATE_KEY	1	x
	Seed	CRYPTO_KE_KEYGENERATE_SEED	16	
	Algorithm	CRYPTO_KE_KEYGENERATE_ALGORITHM	17	
Certificate Parsing	Certificate	CRYPTO_KE_CERTIFICATE_DATA	0	x
	Format	CRYPTO_KE_CERTIFICATE_PARSING_FORMAT	18	
	Current Time	CRYPTO_KE_CERTIFICATE_CURRENT_TIME	19	
	Version	CRYPTO_KE_CERTIFICATE_VERSION	20	
	Serial Number	CRYPTO_KE_CERTIFICATE_SERIALNUMBER	21	
	Signature Algroithm	CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM	22	
	Issuer	CRYPTO_KE_CERTIFICATE_ISSUER	23	
	Validity start	CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE	24	
	Validity end	CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER	25	
	Subject	CRYPTO_KE_CERTIFICATE_SUBJECT	26	
	Subject Public Key	CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY	1	
	Extensions	CRYPTO_KE_CERTIFICATE_EXTENSIONS	27	
	Signature	CRYPTO_KE_CERTIFICATE_SIGNATURE	28	

](

The key elements indices of **SWS_Csm_1022** can be extended by the vendor.

[SWS_Csm_00951] [For each key element that contains cryptographic key material, the format of the provided key shall be specified in the configuration used for data exchange, e.g. for `Csm_KeyElementGet()` or `Csm_KeyElementSet()`. The key formats supported by a specific crypto driver are part of the pre-configuration information that comes along with the crypto driver.
](SRS_CryptoStack_00008)

[SWS_Csm_00953] [The following key formats are available:

CRYPTO_KE_FORMAT_BIN_OCTET	Key provided as octet value in binary form[1].
CRYPTO_KE_FORMAT_BIN_SHEKEYS	Combined input/output keys for SHE operation (M1+M2+M3) and (M4+M5).
CRYPTO_KE_FORMAT_BIN_IDENT_PRIVATEKEY_ PKCS8	Private key material in ASN.1 coded form (BER coding) with identification. The data is provided in binary form, not, e.g. as a BASE64 string.
CRYPTO_KE_FORMAT_BIN_IDENT_PUBLICKEY	Public key material in ASN.1 coded form (BER coding) with identification. The data is provided in binary form, not, e.g. as a BASE64 string.
CRYPTO_KE_FORMAT_BIN _RSA_PRIVATEKEY	Private key material in ASN.1 coded form (BER coding). The key material is provided in binary form, not, e.g. as a BASE64 string.
CRYPTO_KE_FORMAT_BIN _RSA_PUBLICKEY	Public key material in ASN.1 coded form (BER coding). The key material is provided in binary form, not, e.g. as a BASE64 string.
CRYPTO_KE_FORMAT_BIN_CERT_X509_V3	TBD
CRYPTO_KE_FORMAT_BIN_CERT_CVC	TBD

A binary Octet is the integer representation in base 256. A large value can be splitted into his factors:
$x = x_{xLen-1} * 256^{xLen-1} + x_{xLen-2} * 256^{xLen-2} + \ldots + x_1 * 256 + x_0$. where $0 <= x_i < 256$.

Let the Octet Xi have the integer value x_{xLen-i} for 1 <= i <= xLen. The octet is then
$X = X_1 X_2 .. X_{xLen}$

Rationale: An asymmetric key can either be provided with or without identification. The identification is used to uniquely identify the key itself that is provided, so that the key parser can check if the key material is appropriate or not. Without identification, the key material must correspond to the format that is specified for this key. Following IETF standards, the identification of a key is provided as an object identifier (OID) as part of the ASN.1 description.
] (SRS_CryptoStack_00008)

[SWS_Csm_00952] [Vendor specific keyElementIds should start 1000 to avoid interferences with future extended versions of the Crypto Stack.
]()

Note:
The key elements `CRYPTO_KE_[…]_ALGORITHM` are used to configure the behavior of the key management functions, because they are independent of jobs and therefore can not be configured like a primitive.

7.2.2.4 Redirection of Input and/or Output of Crypto Jobs

[SWS_Csm_91013] [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.
]()

[SWS_Csm_91014] [If an input or output value of a job is re-directed to a key element (CsmInOutRedirectionRef ECUC_Csm_00262 is existing) and the corresponding input or output length value is not set to 0, the job shall not be processed and E_NOT_OK shall be returned.
]()

[SWS_Csm_91015]⌈If input or output redirection is not used for a job element (no CsmInOutRedirectionRef ECUC_Csm_00262 is existing), jobRedirectionInfoRef shall be set to NULL_PTR. If redirection is used element (CsmInOutRedirectionRef ECUC_Csm_00262 is existing) the jobRedirectionInfoRef shall point to a structure of Crypto_JobRedirectionInfoType.
⌋()

[SWS_Csm_91016] The structure Crypto_JobRedirectionInfoType contains information which key elements shall be used for redirection. A bit field called redirectionConfig is provided that indicates which input and/or output value is redirected.
The value of redirectionConfig is a bit coded value that is used to indicate, which of the input and output buffers are redirected. If the least significant bit (Bit #0 or 0x01) of redirectionConfig is set the primary input key and its element is redirected and the value of inputKeyId and inputKeyElementId must indicate the element that is used for input buffer instead of the inputPtr and its length. If Bit #1 is set, the secondaryInputBuffer is redirected to the secondary input key is set and the key and key elements must be set, and Bit #2 is used for the tertiary input key. Bit #3 is reserved for future use.
If Bit #4 is set the outputPtr is redirected to the output key element of the output key. Bit #5 indicates the redirection of the secondary output buffer to the secondary key and its key element. If a bit is set to 0 the input or output shall not be redirected to the associated Key Element.
Example: A value of redirectionConfig of "00110001" indicates that the input should be gathered from the inputKeyElement of inputKeyId and that the output buffer and secondary output buffer shall be redirected to the outputKeyElement of outputKeyId and secondaryOutputKeyElement of secondaryOutputKeyId.
⌋()

7.3 Error Classification

7.3.1 Development Errors

[SWS_Csm_91004]⌈Development Error Types

Type of error	*Related error code*	*Value [hex]*
API request called with invalid parameter (Nullpointer)	CSM_E_PARAM_POINTER	0x01
Buffer is too small for operation	CSM_E_SMALL_BUFFER	0x03
keyID is out of range	CSM_E_PARAM_HANDLE	0x04
API request called before initialization of CSM module	CSM_E_UNINIT	0x05
Initialization of CSM module failed	CSM_E_INIT_FAILED	0x07
API request called with invalid processing mode	CSM_E_PROCESSING_MODE	0x08

⌋(SRS_CryptoStack_00086)

7.3.2 Runtime Errors

[SWS_Csm_01089][Runtime Error Types

Type of error	*Related error code*	*Value [hex]*
Queue overrun	CSM_E_QUEUE_FULL	0x01

](SRS_CryptoStack_00086)

7.3.3 Transient Faults

There are no transient faults.

7.3.4 Production Errors

There are no production errors.

7.3.5 Extended Production Errors

There are no extended production errors.

7.4 Error detection

[SWS_Csm_91008] [While the CSM is not initialized and any function of the CSM API is called, except of `CSM_Init()` and `Csm_GetVersionInfo()`, the operation shall not be performed and `CSM_E_UNINIT` shall be reported to the DET when `CsmDevErrorDetect` is true.
]()

[SWS_Csm_91009] [If a pointer to null is passed to an API function and the corresponding input or output data are not re-directed to a key element, the operation shall not be performed and `CSM_E_PARAM_POINTER` shall be reported to the DET when `CsmDevErrorDetect` is true.
]()

[SWS_Csm_91011] [If a CSM API with a key handle in its interface is called and the key handle (called keyID) is out of range, the operation shall not be performed and `CSM_E_PARAM_HANDLE` shall be reported to the DET when `CsmDevErrorDetect` is true.
]()

[SWS_Csm_91012] [If a CSM API is called with a buffer too small to perform the desired operation, the operation shall not be performed and `CSM_E_SMALL_BUFFER` shall be reported to the DET when `CsmDevErrorDetect` is true.
](SRS_CryptoStack_00087)

[SWS_Csm_01088] [If a CSM job needs to be queued and the queue is full, the runtime error `CSM_E_QUEUE_FULL` shall be reported to the DET.](SRS_CryptoStack_00087)

Note: The indication of an queue overrun is logged as runtime error.

8 API Specification

8.1 Imported types

[SWS_Csm_00068] ⌈Only the standard AUTOSAR types provided by StandardTypes.h shall be imported.
⌋()

The Crypto Stack API uses the following extension to `Std_ReturnType`:
[SWS_Csm_01069] ⌈

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_EXHAUSTION`	`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:	Csm module specific return values for use in Std_ReturnType.		
Available via:	`Csm.h`		

⌋ (SRS_CryptoStack_00095)

8.2 Type Definitions

8.2.1 Csm_ConfigType

[SWS_Csm_01085] ⌈

Name:	`Csm_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	The content of the configuration data structure is implementation specific.
Description:	Configuration data structure of Csm module	
Available via:	`Csm.h`	

] (SWS_BSW_00216)

8.2.2 Crypto_AlgorithmFamilyType

[SWS_Csm_01047] [

Name:	Crypto_AlgorithmFamilyType		
Type:	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_ECIES	0x1d	ECIES Cipher
	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		

] ()

8.2.3 Crypto_AlgorithmModeType

[SWS_Csm_01048] [

Name:	Crypto_AlgorithmModeType		
Type:	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 Modex
	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
Description:	Enumeration of the algorithm mode		
Available via:	Csm.h		

] ()

8.2.4 Crypto_InputOutputRedirectionConfigType

[SWS_Csm_91024] ⌈

Name:	Crypto_InputOutputRedirectionConfigType		
Type:	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		

⌋ ()

8.2.5 Crypto_JobStateType

[SWS_Csm_01013] ⌈

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

			crypto primitive.
	Crypto_JobRedirectionInfoType*	jobRedirectionInfoRef	Pointer to a structure containing further information on the usage of keys as input and output for jobs.
Description:	Structure which contains further information, which depends on the job and the crypto primitive.		
Available via:	<none>		

] ()

8.2.6 Crypto_JobStateType

[SWS_Csm_01028] [

Name:	Crypto_JobStateType		
Type:	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		

] ()

8.2.7 Crypto_JobPrimitiveInputOutputType

[SWS_Csm_01009] [

Name:	Crypto_JobPrimitiveInputOutputType		
Type:	Structure		
Element:	const uint8*	inputPtr	Pointer to the input data.
	uint32	inputLength	Contains the input length in bytes.
	const uint8*	secondaryInputPtr	Pointer to the secondary input data (for MacVerify, SignatureVerify).
	uint32	secondaryInputLength	Contains the secondary input length in bytes.
	const uint8*	tertiaryInputPtr	Pointer to the tertiary input data (for MacVerify, SignatureVerify).
	uint32	tertiaryInputLength	Contains the tertiary

			input length in bytes.
	uint8*	outputPtr	Pointer to the output data.
	uint32*	outputLengthPtr	Holds a pointer to a memory location containing the output length in bytes.
	uint8*	secondaryOutputPtr	Pointer to the secondary output data.
	uint32*	secondaryOutputLengthPtr	Holds a pointer to a memory location containing the secondary output length in bytes.
	uint64	input64	versatile input parameter
	Crypto_VerifyResultType*	verifyPtr	Output pointer to a memory location holding a Crypto_VerifyResultType
	uint64*	output64Ptr	Output pointer to a memory location holding a uint64.
	Crypto_OperationModeType	mode	Indicator of the mode(s)/operation(s) to be performed
	uint32	cryIfKeyId	Holds the CryIf key id for key operation services.
	uint32	targetCryIfKeyId	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		

] ()

8.2.8 Crypto_JobInfoType

[SWS_Csm_01010] [

Name:	Crypto_JobInfoType		
Type:	Structure		
Element:	const uint32	jobId	The family of the algorithm
	const uint32	jobPriority	Specifies the importance of the job (the higher, the more important).
Description:	Structure which contains job information (job ID and job priority).		
Available via:	<none>		

] (SRS_CryptoStack_00102)

8.2.9 Crypto_JobPrimitiveInfoType

[SWS_Csm_01012] [

Name:	Crypto_JobPrimitiveInfoType		
Type:	Structure		
Element:	uint32	callbackId	Identifier of the callback

			function, to be called, if the configured service finished.
	const Crypto_PrimitiveInfoType*	primitiveInfo	Pointer to a structure containing further configuration of the crypto primitives
	uint32	cryIfKeyId	Identifier of the CryIf key.
	Crypto_ProcessingType	processingType	Determines the synchronous or asynchronous behavior.
	boolean	callbackUpdateNotification	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		

] (SRS_CryptoStack_00008)

8.2.10 **Crypto_ServiceInfoType**

[SWS_Csm_01031] [

Name:	Crypto_ServiceInfoType		
Type:	Enumeration		
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		

] ()

8.2.11 Crypto_JobRedirectionInfoType

[SWS_Csm_91026] [

Name:	Crypto_JobRedirectionInfoType		
Type:	Structure		
Element:	uint8	redirectionConfig	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.
	uint32	inputKeyId	Identifier of the key which shall be used as input
	uint32	inputKeyElementId	Identifier of the key element which shall be used as input
	uint32	secondaryInputKeyId	Identifier of the key which shall be used as secondary input
	uint32	secondaryInputKeyElementId	Identifier of the key element which shall be used as secondary input
	uint32	tertiaryInputKeyId	Identifier of the key which shall be used as tertiary input
	uint32	tertiaryInputKeyElementId	Identifier of the key element which shall be used as tertiary input
	uint32	outputKeyId	Identifier of the key which shall be used as output
	uint32	outputKeyElementId	Identifier of the key element which shall be used as output
	uint32	secondaryOutputKeyId	Identifier of the key which shall be used as secondary output
	uint32	secondaryOutputKeyElementId	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:	--		

] ()

8.2.12 Crypto_AlgorithmInfoType

[SWS_Csm_01008] [

Name:	Crypto_AlgorithmInfoType		
Type:	Structure		
Element:	Crypto_AlgorithmFamilyType	family	The family of the algorithm
	Crypto_AlgorithmFamilyType	secondaryFamily	The secondary family of the algorithm
	uint32	keyLength	The key length in bits to be

			used with that algorithm
	Crypto_AlgorithmModeType	mode	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		

] ()

8.2.13 Crypto_ProcessingType

[SWS_Csm_01049] [

Name:	Crypto_ProcessingType		
Type:	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		

] (SRS_CryptoStack_00100, SRS_CryptoStack_00101)

8.2.14 Crypto_PrimitiveInfoType

[SWS_Csm_01011] [

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

] ()

8.2.15 Csm_ConfigIdType

[SWS_Csm_00691] [

Name:	Csm_ConfigIdType		
Type:	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		

] (SRS_Csm_00066)

8.3 Function Definitions

[SWS_Csm_00478] ⌈All functions need not to be reentrant. For behavior in case of a reentrant call see **SWS_Csm_00017**.
⌋()

8.3.1 General Interface

8.3.1.1 Csm_Init

[SWS_Csm_00646] ⌈

Service name:	Csm_Init	

Syntax:	`void Csm_Init(`
	`  const Csm_ConfigType* configPtr`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	configPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the CSM module.	
Available via:	`Csm.h`	

⌋ (SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)

[SWS_Csm_00186] ⌈The Configuration pointer `configPtr` shall always have a null pointer value.
⌋(SWS_BSW_00050)

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

[SWS_Csm_00659] ⌈If the initialization of the CSM module fails, the CSM shall report `CSM_E_INIT_FAILED` to the DET when `CsmDevErrorDetect` is true.
⌋()

8.3.1.2 Csm_GetVersionInfo

[SWS_Csm_00705] ⌈

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.	
Available via:	`Csm.h`	

] (SRS_BSW_00407)

8.3.2 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.

8.3.2.1 Csm_Hash

[SWS_Csm_00980] [

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:	Sync or Async, 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 perfom.
	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 CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result
Description:	Uses the given data to perform the hash calculation and stores the hash.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00024)

8.3.3 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.

8.3.3.1 Csm_MacGenerate

[SWS_Csm_00982] ⌈

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:	Sync or Async, dependent 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 perfom.
	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_SMALL_BUFFER:The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

⌋ (SRS_CryptoStack_00022)

8.3.3.2 Csm_MacVerify

[SWS_Csm_01050] ⌈

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:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Indicates which operation mode(s) to perfom.
	mode	Indicates which operation mode(s) to perfom.
	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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] ()

8.3.4 Cipher Interface

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

8.3.4.1 Csm_Encrypt

[SWS_Csm_00984] [

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:	Sync or Async, dependend 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 perfom.
	dataPtr	Contains the pointer to the data to be encrypted.
	dataLength	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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	Csm.h	

] (SRS_CryptoStack_00020, SRS_CryptoStack_00021)

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.

8.3.4.2 **Csm_Decrypt**

[SWS_Csm_00989] [

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:	Sync or Async, dependend 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 perfom.
	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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00020, SRS_CryptoStack_00021)

8.3.5 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).

8.3.5.1 Csm_AEADEncrypt

[SWS_Csm_01023] [

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:	Sync or Async, dependend 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 perfom.
	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	ciphertextLengthPtr	Holds a pointer to the memory location in which the output

(inout):		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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] ()

8.3.5.2 **Csm_AEADDecrypt**

[SWS_Csm_01026] [

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:	Sync or Async, dependend 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 perfom.
	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 paintext 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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] ()

8.3.6 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.

8.3.6.1 Csm_SignatureGenerate

[SWS_Csm_00992] [

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:	Sync or Async, dependend 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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00023)

8.3.6.2 **Csm_SignatureVerify**

[SWS_Csm_00996] [

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:	Sync or Async, dependend 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_SMALL_BUFFER: The provided buffer is too small to store the result CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00023)

8.3.7 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.

8.3.7.1 Csm_RandomGenerate

[SWS_Csm_01543] [

Service name:	Csm_RandomGenerate	

Syntax:	`Std_ReturnType Csm_RandomGenerate(`
	`  uint32 jobId,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr`
	`)`	

Service ID[hex]:	0x72	
Sync/Async:	Sync or Async, dependend 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_EXHAUSTION: 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.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00019)

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

[SWS_Csm_01054] [The operation mode of the Csm_RandomGenerate() function call shall be set to ”CRYPTO_OPERATIONMODE_SINGLECALL”.
](()

8.3.8 Key Management Interface

The following interfaces are used for key management. Basically, a key contains of one ore 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)

[SWS_Csm_00974] [If a key management function is called, the CSM shall disable processing new jobs from the queue until the next call of the main function.
](()

8.3.8.1 Key Setting Interface

8.3.8.1.1 Csm_KeyElementSet

[SWS_Csm_00957] [

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.	
Available via:	`Csm.h`	

] ()

[SWS_Csm_01002] [If no errors are detected by Csm, the service `Csm_KeyElementSet()` shall call `CryIf_KeyElementSet()`.
] ( )

8.3.8.1.2 Csm_KeySetValid

[SWS_Csm_00958] [

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, Crypro Driver Object is busy
Description:	Sets the key state of the key identified by keyId to valid.	
Available via:	`Csm.h`	

] ()

[SWS_Csm_01003] [If no errors are detected by Csm, the service `Csm_KeySetValid()` shall call `CryIf_KeySetValid()`.
] ( )

8.3.8.2 Key Extraction Interface

8.3.8.2.1 Csm_KeyElementGet

[SWS_Csm_00959] [

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.
	keyElementId	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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00010, SRS_CryptoStack_00011, SRS_CrytptoStack_00029)

[SWS_Csm_01004] [If no errors are detected by Csm, the service `Csm_KeyElementGet()` shall call `CryIf_KeyElementGet()`.
] ( )

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

8.3.8.3 Key Copying Interface

8.3.8.3.1 Csm_KeyElementCopy

[SWS_Csm_00969] [

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.	
Available via:	`Csm.h`	

] ()

[SWS_Csm_01032] [If no errors are detected by Csm and the `keyId` and `targetKeyId` are located in different Crypto Drivers, the service `Csm_KeyElementCopy()` shall call `CryIf_KeyElementCopy()` and pass on the return value.
] ( )

8.3.8.3.2 Csm_KeyCopy

[SWS_Csm_01034] [

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.	

Available via:	`Csm.h`

] ()

[SWS_Csm_01035] [If no errors are detected by Csm and the `keyId` and `targetKeyId` are located in the same Crypto Driver, the service `Csm_KeyCopy()` shall call `CryIf_KeyCopy()` and pass on the return value.
] ( )

8.3.8.3.3 Csm_KeyElementCopyPartial

[SWS_Csm_91025] [

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.
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.

[SWS_Csm_91019] [If no errors are detected by Csm shall call `CryIf_KeyElementCopyPartial()` and pass on the return value.
] ( )

[SWS_Csm_91020] [If the current length of the target key element is greater or equal than (keyElementTargetOffset + keyElementCopyLength), the key element length remains unchanged and the target data is overwritten with the contents of the source data.
] ( )

[SWS_Csm_91021] [If the current length of the target key element is lower than (`keyElementTargetOffset` + `keyElementCopyLength`) and the maximum length of the key element is greater or equal than (`keyElementTargetOffset` + `keyElementCopyLength`), then the source data shall be copied into the target key element and the length shall be set to (`keyElementTargetOffset` + `keyElementCopyLength`).
] ( )

[SWS_Csm_91022] [
If the maximum length of the target key element is lower than (`keyElementTargetOffset` + `keyElementCopyLength`) then the copy operation shall not be performed and the function shall return with the error code `CRYPTO_E_KEY_SIZE_MISMATCH`.
] ( )

8.3.8.4 Key Generation interface

8.3.8.4.1 Csm_RandomSeed

[SWS_Csm_01051] [

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.	
Available via:	`Csm.h`	

] ()

[SWS_Csm_01052] [If no errors are detected by Csm, the service `Csm_RandomSeed()` shall call `CryIf_RandomSeed()`.
] ( )

8.3.8.4.2 Csm_KeyGenerate

[SWS_Csm_00955] [

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 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.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00026, SRS_CryptoStack_00027)

[SWS_Csm_01005] [If no errors are detected by Csm, the service `Csm_KeyGenerate()` shall call `CryIf_KeyGenerate()`.
] ( )

8.3.8.5 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.

8.3.8.5.1 Csm_KeyDerive

[SWS_Csm_00956] [

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 targetCryptoKeyId.	
Available via:	`Csm.h`	

] (SRS_CryptoStack_00103) **Csm_KeyGenerate**

[SWS_Csm_01018] [If no errors are detected by Csm, the service `Csm_KeyDerive()` shall call `CryIf_KeyDerive()`.
] ( )

[SWS_Csm_01019] [If the number of iterations for the key derivation is needed by the Crypto Driver, it shall be stored in the key element `CRYPTO_KE_KEYDERIVATION_ITERATIONS`.
] ( )

8.3.8.6 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

8.3.8.6.1 Csm_KeyExchangeCalcPubVal

[SWS_Csm_00966] [

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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] (SRS_CrytptoStack_00028)

[SWS_Csm_01020] [If no errors are detected by Csm, the service `Csm_KeyExchangeCalcPubVal()` shall call `CryIf_KeyExchangeCalcPubVal()`.
] ( )

8.3.8.6.2 Csm_KeyExchangeCalcSecret

[SWS_Csm_00967] [

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_SMALL_BUFFER: The provided buffer is too small to store the result 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.	
Available via:	`Csm.h`	

] (SRS_CrytptoStack_00028)

[SWS_Csm_01006] [If no errors are detected by Csm, the service `Csm_KeyExchangeCalcSecret()` shall call `CryIf_KeyExchangeCalcSecret()`.
] ( )

8.3.8.7 Certificate Interface

8.3.8.7.1 Csm_CertificateParse

[SWS_Csm_01036] [

Service name:	Csm_CertificateParse	
Syntax:	`Std_ReturnType Csm_CertificateParse( const uint32 keyId )`	
Service ID[hex]:	0x6e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key to be used for the certificate parsing.
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:	This function shall dispatch the certificate parse function to the CRYIF.
Available via:	`Csm.h`

] (SRS_CrytptoStack_00031)

[SWS_Csm_01037] [If no errors are detected by Csm, the service `Csm_CertificateParse()` shall call `CryIf_CertificateParse()`.
] ( )

8.3.8.7.2 Csm_CertificateVerify

[SWS_Csm_01038] [

Service name:	Csm_CertificateVerify	

Syntax:	`Std_ReturnType Csm_CertificateVerify(`
	`  const uint32 keyId,`
	`  const uint32 verifyCryptoKeyId,`
	`  Crypto_VerifyResultType* verifyPtr`
	`)`	

Service ID[hex]:	0x74	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant but not for the same cryptoKeyId	
Parameters (in):	keyId	Holds the identifier of the key which shall be used to validate the certificate.
	verifyCryptoKeyId	Holds the identifier of the key containing the certificate to be verified.
Parameters (inout):	None	
Parameters (out):	verifyPtr	Holds a pointer to the memory location which will contain the result of the certificate verification.
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:	Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId. Note: Only certificates stored in the same Crypto Driver can be verified against each other. If the key element CRYPTO_KE_CERTIFICATE_CURRENT_TIME is used for the verification of the validity period of the certificate indentified by verifyKeyId, it shall have the same format as the timestamp in the certificate.	
Available via:	`Csm.h`	

] ()

[SWS_Csm_01040] [If no errors are detected by Csm, the service `Csm_CertificateVerify ()` shall call `CryIf_CertificateVerify()`.
] ( )

8.3.9 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.

8.3.9.1 Csm_JobKeySetValid

[SWS_Csm_91027] [

Service name:	Csm_JobKeySetValid	
Syntax:	`Std_ReturnType Csm_JobKeySetValid( uint32 jobId, uint32 keyId )`	
Service ID[hex]:	0x7a	
Sync/Async:	Sync or Async, 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, Crypro Driver Object is busy
Description:	Stores the key if necessary and sets the key state of the key identified by keyId to valid.	
Available via:	`Csm.h`	

] ()

8.3.9.2 Csm_JobRandomSeed

[SWS_Csm_91028] [

Service name:	Csm_JobRandomSeed	
Syntax:	`Std_ReturnType Csm_JobRandomSeed( uint32 jobId, uint32 keyId, const uint8* seedPtr, uint32 seedLength )`	
Service ID[hex]:	0x7b	
Sync/Async:	Sync or Async, 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_QUEUE_FULL: Request failed, the queue is full 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.	

Available via:	`Csm.h`

] ()

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

8.3.9.3 Csm_JobKeyGenerate

[SWS_Csm_91029] [

Service name:	Csm_JobKeyGenerate	
Syntax:	`Std_ReturnType Csm_JobKeyGenerate( uint32 jobId, uint32 keyId )`	
Service ID[hex]:	0x7c	
Sync/Async:	Sync or Async, 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_QUEUE_FULL: Request failed, the queue is full 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.	
Available via:	`Csm.h`	

] ()

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

8.3.9.4 Csm_JobKeyDerive

[SWS_Csm_91030] [

Service name:	Csm_JobKeyDerive	
Syntax:	`Std_ReturnType Csm_JobKeyDerive( uint32 jobId, uint32 keyId, uint32 targetKeyId )`	
Service ID[hex]:	0x7d	
Sync/Async:	Sync or Async, 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_QUEUE_FULL: Request failed, the queue is full 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 targetCryptoKeyId.	
Available via:	`Csm.h`	

] ()

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

8.3.9.5 **Csm_JobKeyExchangeCalcPubVal**

[SWS_Csm_91031] [

Service name:	Csm_JobKeyExchangeCalcPubVal	

Syntax:	`Std_ReturnType Csm_JobKeyExchangeCalcPubVal(`
	`  uint32 jobId,`
	`  uint32 keyId,`
	`  uint8* publicValuePtr,`
	`  uint32* publicValueLengthPtr`
	`)`	

Service ID[hex]:	0x7e	
Sync/Async:	Sync or Async, 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.
	publicValuePtr	Contains the pointer to the data where the public value shall be stored.
Parameters (inout):	None	
Parameters (out):	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.
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result CRYPTO_E_QUEUE_FULL: Request failed, the queue is full 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.	
Available via:	`Csm.h`	

] ()

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

8.3.9.6 **Csm_JobKeyExchangeCalcSecret**

[SWS_Csm_91032] [

Service name:	Csm_JobKeyExchangeCalcSecret	

Syntax:	`Std_ReturnType Csm_JobKeyExchangeCalcSecret(`
	`  uint32 jobId,`
	`  uint32 keyId,`
	`  const uint8* partnerPublicValuePtr,`
	`  uint32 partnerPublicValueLength`
	`)`	

Service ID[hex]:	0x7f	
Sync/Async:	Sync or Async, 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 E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result CRYPTO_E_QUEUE_FULL: Request failed, the queue is full 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.	
Available via:	`Csm.h`	

] ()

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

8.3.9.7 Csm_JobCertificateParse

[SWS_Csm_91033] [

Service name:	Csm_JobCertificateParse	
Syntax:	`Std_ReturnType Csm_JobCertificateParse( uint32 jobId, uint32 keyId )`	
Service ID[hex]:	0x80	
Sync/Async:	Sync or Async, 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 to be used for the certificate parsing.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_QUEUE_FULL: Request failed, the queue is full 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:	This function shall dispatch the certificate parse function to the CRYIF.	
Available via:	`Csm.h`	

] ()

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

8.3.9.8 Csm_JobCertificateVerify

[SWS_Csm_91034] [

Service name:	Csm_JobCertificateVerify	
Syntax:	`Std_ReturnType Csm_JobCertificateVerify( const uint32 jobId, const uint32 keyId, const uint32 verifyKeyId, Crypto_VerifyResultType* verifyPtr )`	
Service ID[hex]:	0x81	
Sync/Async:	Sync or Async, 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 to validate the certificate.
	verifyKeyId	Holds the identifier of the key containing the certificate to be verified.
Parameters (inout):	None	
Parameters (out):	verifyPtr	Holds a pointer to the memory location which will contain the result of the certificate verification.
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed

		E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_QUEUE_FULL: Request failed, the queue is full 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:	Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId. Note: Only certificates stored in the same Crypto Driver can be verified against each other. If the key element CRYPTO_KE_CERTIFICATE_CURRENT_TIME is used for the verification of the validity period of the certificate indentified by verifyKeyId, it shall have the same format as the timestamp in the certificate.	
Available via:	`Csm.h`	

] ()

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

8.3.10 Job Cancellation Interface

8.3.10.1 Csm_CancelJob

[SWS_Csm_00968] [

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 finish callback.
Description:	Cancels the job processing from asynchronous or streaming jobs.	
Available via:	`Csm.h`	

] ()

[SWS_Csm_01086] [If development error detection for the CSM is enabled: The function `Csm_CancelJob()` shall raise the error `CSM_E_PROCESSING_MODE` and return `E_NOT_OK` if the `Csm_CancelJob()` is called for a synchronous job.
] ( )

[SWS_Csm_01021] [The Csm shall call `CryIf_CancelJob()` to cancel a potential job in the driver.
Further the CSM shall remove the job from its own queue.
] ( )

[SWS_Csm_01030] [In case the `CryIf_CancelJob()` returns E_OK, the job finish callback CallbackNotification shall be called with a result value of `E_JOB_CANCELED`.
] ( )

[SWS_Csm_01087] [In case the `CryIf_CancelJob()` returns `CRYPTO_E_JOB_CANCELED` (i.e. the job was not instantly cancled) the CSM shall postpone the call of the job finish callback until the next call of `Csm_CallbackNotification()`. The result of the job finish callback shall be `E_JOB_CANCELED`.
] ( )

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

8.3.11 Callback Notifications

8.3.11.1 Csm_CallbackNotification

[SWS_Csm_00970] [

Service name:	Csm_CallbackNotification	
Syntax:	`void Csm_CallbackNotification( Crypto_JobType* job, Csm_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). Variation: {ecuc(Csm/CsmJob/CsmJobUsePort == false)} && {ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef->CsmPrimitives/{Primitive}Config/{Primitive}Processing == CRYPTO_PROCESSING_ASYNC)}	
Available via:	`Csm.h`	

] (SRS_BSW_00359, SRS_BSW_00360)

[SWS_Csm_01053] [If the `CRYPTO_OPERATIONMODE_UPDATE` bit is set in `job->jobPrimitiveInputOutput.mode` and the corresponding `CsmJobPrimitiveCallbackUpdateNotification` (**ECUC_Csm_00124**) is true, the `Csm_CallbackNotification` shall call the configured callback function.
](

[SWS_Csm_01044][If the `CRYPTO_OPERATIONMODE_FINISH` bit is set in `job->jobPrimitiveInputOutput.mode`, the `Csm_CallbackNotification` shall call the configured callback function.
](

[SWS_Csm_91017][If the `CRYPTO_OPERATIONMODE_FINISH` bit is set in `job->jobPrimitiveInputOutput.mode` and `CsmProcessingMode` is set to `CRYPTO_PROCESSING_ASYNC` and `CsmJobInterfaceUsePort` is set to CRYPTO_USE_PORT_OPTIMIZED, the `CSM` shall trigger CallbackNotification service.
](

8.3.12 Scheduled functions

8.3.12.1 Csm_MainFunction

[SWS_Csm_00479] [

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.
Available via:	`SchM_Csm.h`

] (SRS_BSW_00373, SRS_BSW_00432)

8.4 Expected Interfaces

8.4.1 Interfaces to Standard Software Modules

[SWS_Csm_00484] [In this section, all interfaces required from other modules are listed.
](

[SWS_Csm_00485] [The CSM module shall use an AUTOSAR Det module for development error notification.
](

8.4.2 Mandatory Interfaces

API function	*Description*
CryIf_ProcessJob	
CryIf_CancelJob	
CryIf_KeyElementSet	
CryIf_KeySetValid	
CryIf_KeyElementGet	
CryIf_KeyElementCopy	
CryIf_KeyCopy	
CryIf_RandomSeed	
CryIf_KeyGenerate	
CryIf_KeyExchangeCalcSecret	
CryIf_CertificateParse	
CryIf_CertificateVerify	

8.4.3 Optional Interfaces

API function	*Header File*	*Description*

8.4.4 Configurable interfaces

8.4.4.1 Csm_ApplicationCallbackNotification

[SWS_Csm_00971] [

Service name:	Csm_ApplicationCallbackNotification	
Syntax:	`void Csm_ApplicationCallbackNotification( const uint32 jobID, Csm_ResultType result )`	
Service ID[hex]:	0x80	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	jobID	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.	
Available via:	`Csm.h`	

] (SRS_BSW_00359, SRS_BSW_00360)

8.5 Service Interface

This chapter is an addition to the specification of the Csm module. Whereas the other parts of the specification define the behavior and the C-interfaces of the corresponding basic software module, this chapter formally specifies the corresponding AUTOSAR service in terms of the SWC template. The interfaces

described here will be visible on the VFB and are used to generate the RTE between application software and the Csm module.

8.5.1 Client-Server-Interfaces

8.5.1.1 CsmKeyManagement_{Key}

[SWS_Csm_01905] [

Name	CsmKeyManagement_{Key}	
Comment	Interface to execute the key management functions.	
IsService	true	
Variation	({ecuc(Csm/CsmKeys/CsmKey.CsmKeyUsePort)} == TRUE) Key = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	6	CSM_E_KEY_READ_FAIL
	7	CSM_E_KEY_WRITE_FAIL
	8	CSM_E_KEY_NOT_AVAILABLE
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

CertificateParse		
Comments	This function shall dispatch the certificate parse function to the CRYIF.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_BUSY	Request failed, service is still busy.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.
CertificateVerify		

Comments	Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId. Note: Only certificates stored in the same Crypto Driver can be verified against each other. If the key element CRYPTO_KE_CERTIFICATE_CURRENT_TIME is used for the verification of the validity period of the certificate indentified by verifyKeyId, it shall have the same format as the timestamp in the certificate		
Variation	--		
Parameters	verifyKeyId	Comment	Holds the identifier of the key containing the certificate to be verified
		Type	uint32
		Variation	--
		Direction	IN
	verifyPtr	Comment	Contains the result of the certificate verification
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

KeyCopy			
Comments	This function shall copy all key elements from the source key to a target key.		
Variation	--		
Parameters	targetKeyId	Comment	Holds the identifier of the key whose key element shall be the destination element.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	

	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

KeyDerive			
Comments	Derives a new key by using the key elements in the given key. 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 targetCryptoKeyId.		
Variation	--		
Parameters	targetKeyId	Comment	Holds the identifier of the key which is used to store the derived key.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

KeyElementCopy	
Comments	This function shall copy a key elements from one key to a target key

Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element which shall be the source for the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyId	Comment	Holds the identifier of the key whose key element shall be the destination element.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyElementId	Comment	Holds the identifier of the key element which shall be the destination for the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

KeyElementCopyPartial	
Comments	This function shall copy parts of a a key elements from one key to parts of a target key element of a target key.

Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element which shall be the source for the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
	keyElementSourceOffset	Comment	This is the offset of the source key element indicating the start index of the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
	keyElementTargetOffset	Comment	This is the offset of the destination key element indicating the start index of the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
	keyElementCopyLength	Comment	Specifies the number of bytes that shall be copied.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyId	Comment	Holds the identifier of the key whose key element shall be the destination element.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyElementId	Comment	Holds the identifier of the key element which shall be the destination for the copy operation.
		Type	uint32
		Variation	--
		Direction	IN

Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_BUSY	Request failed, service is still busy.
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

KeyElementGet			
Comments	Retrieves the key element bytes from a specific key element of the key and stores the key element in the provided buffer.		
Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element to be read.
		Type	uint32
		Variation	--
		Direction	IN
	keyPtr	Comment	Holds the data to the key element bytes to be written.
		Type	Csm_KeyDataType_{Crypto}
		Variation	--
		Direction	OUT
	keyLength	Comment	Contains the number of key element bytes.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

	CSM_E_BUSY	Request failed, service is still busy.
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

KeyElementSet			
Comments	Sets the given key element bytes to the key.		
Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element to be written.
		Type	uint32
		Variation	--
		Direction	IN
	keyPtr	Comment	Holds the data to the key element bytes to be processed.
		Type	Csm_KeyDataType_{Crypto}
		Variation	--
		Direction	IN
	keyLength	Comment	Contains the number of key element bytes.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not	

		partially accessible and the provided key element length is too short or too long for that key element.	

KeyExchangeCalcPubVal			
Comments	Calculates the public value of the current user for the key exchange and stores the public key in the provided buffer		
Variation	--		
Parameters	publicValuePtr	Comment	Holds a pointer to the memory location in which the public value length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	publicValueLengthPtr	Comment	Contains the pointer to the data where the public value shall be stored.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

KeyExchangeCalcSecret			
Comments	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.		
Variation	--		
Parameters	partnerPublicValuePtr	Comment	Holds the pointer to the memory location containing the partner's public value

		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	partnerPublicValueLength	Comment	Contains the number of bytes of the partner pulic value
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

KeyGenerate		
Comments	Generates new key material and store it in the key identified by keyId.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_BUSY	Request failed, service is still busy.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

KeySetValid		
Comments	Sets the given key element bytes to the key.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_BUSY	Request failed, service is still busy.

RandomSeed

Comments	Feeds the key element CRYPTO_KE_RANDOM_SEED with a random seed.		
Variation	--		
Parameters	seedPtr	Comment	Holds the data which shall be used for the random seed initialization.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	seedLength	Comment	Contains the length of the seed in bytes.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	

] (SRS_Csm_00066)

8.5.1.2 **CsmHash_{PrimitiveCfg}**

[SWS_Csm_00946] [

Name	CsmHash_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the hash calculation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

Hash	
Comments	Streaming approach of the hash calculation.

Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be hashed.
		Type	Csm_HashDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be hashed.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the hash.
		Type	Csm_HashResultType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the hash.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	

⌋ (SRS_CryptoStack_00090)

8.5.1.3 **CsmMacGenerate_{PrimitiveCfg}**

[SWS_Csm_09000] ⌈

Name	CsmMacGenerate_{PrimitiveCfg}
Comment	Synchronous processing interface to execute the MAC generation.
IsService	true
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-

	NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

MacGenerate			
Comments	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data from which a MAC shall be generated of.
		Type	Csm_MacGenerateDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the MAC.
		Type	Csm_MacGenerateResultType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the MAC.

		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.1.4 **CsmMacVerify_{PrimitiveCfg}**

[SWS_Csm_00936] [

Name	CsmMacVerify_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the MAC verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

MacVerify	
Comments	Uses the given data to perform a MAC generation and stores the MAC in the memory

	location pointed to by the MAC pointer.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data from which a MAC shall be generated of.
		Type	Csm_MacVerifyDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for whichs MAC shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
	compareBuffer	Comment	Contains the MAC to be verified.
		Type	Csm_MacVerifyCompareType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	compareLength	Comment	Contains the length in BITS of the MAC to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the verification result.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element	

		length is too short or too long for that key element.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

] (SRS_CryptoStack_00090)

8.5.1.5 CsmEncrypt_{PrimitiveCfg}

[SWS_Csm_00947] [

Name	CsmEncrypt_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the encryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

Encrypt			
Comments	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be encrypted.
		Type	Csm_EncryptDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32

		Variation	--
		Direction	IN
	result	Comment	Contains the data of the cipher.
		Type	Csm_EncryptResultType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the cipher.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00906)

8.5.1.6 **CsmDecrypt_{PrimitiveCfg}**

[SWS_Csm_01906] [

Name	CsmDecrypt_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the decryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

Decrypt			
Comments	Streaming approach of the decryption.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be decrypted.
		Type	Csm_DecryptDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be decrypted.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the decrypted plaintext.
		Type	Csm_DecryptResultType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the decrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	

	CSM_E_BUSY	Request failed, service is still busy.
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

] (SRS_CryptoStack_00090)

8.5.1.7 **CsmAEADEncrypt_{PrimitiveCfg}**

[SWS_Csm_01910] [

Name	CsmAEADEncrypt_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the AEAD encryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

AEADEncrypt			
Comments	Streaming approach of the AEAD encryption.		
Variation	--		
Parameters	plaintextBuffer	Comment	Contains the plaintext to be encrypted with AEAD.
		Type	Csm_AEADEncryptPlaintextType_{Crypto}
		Variatio	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-

		n	NAME}
		Direction	IN
	plaintextLength	Comment	This element Contains the length in bytes of the plaintext to be encrypted with AEAD.
		Type	uint32
		Variation	--
		Direction	IN
	associatedDataBuffer	Comment	Contains the data of the header (that is not part of the encryption but authentication).
		Type	Csm_AEADEncryptAssociatedDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	associatedDataLength	Comment	Contains the length in bytes of the data of the header.
		Type	uint32
		Variation	--
		Direction	IN
	ciphertextBuffer	Comment	Contains the data of the AEAD cipher.
		Type	Csm_AEADEncryptCiphertextType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	ciphertextLengthPtr	Comment	Contains the length in bytes of the data of the AEAD cipher.
		Type	uint32
		Variation	--
		Direction	INOUT

	tagBuffer	Comment	Contains the data of the Tag.
		Type	Csm_AEADEncryptTagType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	tagLength	Comment	Contains the length in bytes of the data of the Tag.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.1.8 **CsmAEADDecrypt_{PrimitiveCfg}**

[SWS_Csm_01915] [

Name	CsmAEADDecrypt_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the AEAD decryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

AEADDecrypt			
Comments	Streaming approach of the AEAD decryption.		
Variation	--		
Parameters	ciphertextBuffer	Comment	Contains the ciphertext to be decrypted with AEAD.
		Type	Csm_AEADDecryptCiphertextType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	ciphertextLength	Comment	Contains the length in bytes of the ciphertext to be decrypted with AEAD.
		Type	uint32
		Variation	--
		Direction	IN
	associatedDataBuffer	Comment	Contains the data of the header (that is not part of the encryption but authentication) .
		Type	Csm_AEADDecryptAssociatedDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	associatedDataLength	Comment	Contains the length in bytes of the data of the header.
		Type	uint32
		Variatio	--

		n	
		Direction	IN
	tagBuffer	Comment	Contains the data of the Tag.
		Type	Csm_AEADDecryptTagType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	tagLength	Comment	Contains the length in BITS of the data of the Tag.
		Type	uint32
		Variation	--
		Direction	IN
	plaintextBuffer	Comment	Contains the data of the decrypted AEAD plaintext.
		Type	Csm_AEADDecryptPlaintextType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	plaintextLength	Comment	Contains the length in bytes of the data of the decrypted AEAD plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
	resultBuffer	Comment	Contains the verification result.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible	E_OK	Operation successful	

Errors	E_NOT_OK	Operation failed
	CSM_E_BUSY	Request failed, service is still busy.
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

] (SRS_CryptoStack_00090)

8.5.1.9 CsmSignatureGenerate_{PrimitiveCfg}

[SWS_Csm_00903] [

Name	CsmSignatureGenerate_{PrimitiveCfg}	
Comment	Synchronous processing interface to generate a signature.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

SignatureGenerate			
Comments	Streaming approach of the signature generation.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data from which the signature shall be generated.
		Type	Csm_SignatureGenerateDataType_{Crypto}

		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which the signature shall be generated.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the signature.
		Type	Csm_SignatureGenerateResultType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the signature.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.1.10 CsmSignatureVerify_{PrimitiveCfg}

[SWS_Csm_00943] [

Name	CsmSignatureVerify_{PrimitiveCfg}

Comment	Synchronous processing interface to execute the signature verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	13	CSM_E_KEY_EMPTY

Operations

SignatureVerify			
Comments	Interface to verify a signature.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data for whichs signature shall be verified.
		Type	Csm_SignatureVerifyDataType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for whichs signature shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
	compareBuffer	Comment	Contains the signature to be verified.
		Type	Csm_SignatureVerifyCompareType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	IN

	compareLength	Comment	Contains the length in bytes of the signature to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the verification result.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.1.11 CsmRandomGenerate_{PrimitiveCfg}

[SWS_Csm_00902] [

Name	CsmRandomGenerate_{PrimitiveCfg}	
Comment	Synchronous processing interface to execute the random number generation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	4	CSM_E_ENTROPY_EXHAUSTION

Operations

RandomGenerate			
Comments	Synchronous processing interface to execute the random number generation.		
Variation	--		
Parameters	resultBuffer	Comment	Contains the random number
		Type	Csm_RandomGenerateResultType_{Crypto}
		Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the data of random number.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_ENTROPY_EXHAUSTION	Request failed, entropy of random number generator is exhausted.	

] (SRS_CryptoStack_00090)

8.5.2 Client-Server-Interfaces (DATA_REFERENCES)

8.5.2.1 CsmHash

[SWS_Csm_91051] [

Name	CsmHash	
Comment	Asynchronous processing interface to execute the hash calculation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	12	CSM_E_JOB_CANCELED

Operations

CancelJob			
Comments	Cancels the job.		
Variation	--		
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.	
	E_NOT_OK	Request failed, job couldn't be removed.	
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.	
Hash			
Comments	Utilize the random seed service.		
Variation	--		
Parameters	dataBuffer	Comment	References the data to be hashed.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be hashed.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	References the data of the hash.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	resultLength	Comment	Contains the length in bytes of the hash.
		Type	uint32

		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	

] (SRS_CryptoStack_00090)

8.5.2.2 **CsmMacGenerate**

[SWS_Csm_91052] [

Name	CsmMacGenerate	
Comment	Asynchronous processing interface to execute the MAC generation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

MacGenerate			
Comments	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.		
Variation	--		
Parameters	dataBuffer	Comment	References the data from which a MAC shall be generated of.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	References the data of the MAC.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the MAC.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.2.3 **CsmMacVerify**

[SWS_Csm_91053] [

Name	CsmMacVerify	
Comment	Asynchronous processing interface to execute the MAC verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

MacVerify			
Comments	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.		
Variation	--		
Parameters	dataBuffer	Comment	References the data from which a

			MAC shall be generated of.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for whichs MAC shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
	compareBuffer	Comment	References the MAC to be verified.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	compareLength	Comment	Contains the length in BITS of the MAC to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the verification result.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.2.4 **CsmEncrypt**

[SWS_Csm_91054] ⌈

Name	CsmEncrypt	
Comment	Asynchronous processing interface to execute the encryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

Encrypt			
Comments	Encrypts the given data and stores the ciphertext in the memory location pointed by the result pointer.		
Variation	--		
Parameters	dataBuffer	Comment	References the data to be encrypted.
		Type	Csm_DataPtr
		Variation	--

		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	result	Comment	References the data of the cipher.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the cipher.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.2.5 **CsmDecrypt**

[SWS_Csm_91055] [

Name	CsmDecrypt	
Comment	Asynchronous processing interface to execute the decryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

Decrypt			
Comments	Decrypts the given data and stores the plaintext in the memory location pointed by the resultBuffer pointer.		
Variation	--		
Parameters	dataBuffer	Comment	References the data to be decrypted.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be decrypted.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	References the data of the decrypted plaintext.

		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the decrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] (SRS_CryptoStack_00090)

8.5.2.6 **CsmAEADEncrypt**

[SWS_Csm_91056] [

Name	CsmAEADEncrypt	
Comment	Asynchronous processing interface to execute the AEAD encryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

AEADEncrypt			
Comments	Streaming approach of the AEAD encryption.		
Variation	--		
Parameters	plaintextBuffer	Comment	References the plaintext to be encrypted with AEAD.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	plaintextLength	Comment	This element Contains the length in bytes of the plaintext to be encrypted with AEAD.
		Type	uint32
		Variation	--
		Direction	IN
	associatedDataBuffer	Comment	References the data of the header (that is not part of the encryption but authentication).
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	associatedDataLength	Comment	Contains the length in bytes of the data of the header.
		Type	uint32
		Variation	--
		Direction	IN
	ciphertextBuffer	Comment	References the data of the AEAD cipher.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	ciphertextLengthPtr	Comment	Contains the length in bytes of the data of the AEAD cipher.
		Type	uint32
		Variation	--

		Direction	INOUT
	tagBuffer	Comment	References the data of the Tag.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	tagLength	Comment	Contains the length in bytes of the data of the Tag.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

] (SRS_CryptoStack_00090)

8.5.2.7 **CsmAEADDecrypt**

[SWS_Csm_91057] [

Name	CsmAEADDecrypt

Comment	Asynchronous processing interface to execute the AEAD decryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

AEADDecrypt			
Comments	Streaming approach of the AEAD decryption.		
Variation	--		
Parameters	ciphertextBuffer	Comment	References the ciphertext to be decrypted with AEAD.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	ciphertextLength	Comment	Contains the length in bytes of the ciphertext to be decrypted with AEAD.
		Type	uint32
		Variation	--
		Direction	IN
	associatedDataBuffer	Comment	References the data of the header (that is not part of the encryption but authentication).
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	associatedDataLength	Comment	Contains the length in bytes of the

			data of the header.
		Type	uint32
		Variation	--
		Direction	IN
	tagBuffer	Comment	References the data of the Tag.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	tagLength	Comment	Contains the length in BITS of the data of the Tag.
		Type	uint32
		Variation	--
		Direction	IN
	plaintextBuffer	Comment	References the data of the decrypted AEAD plaintext.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	plaintextLength	Comment	Contains the length in bytes of the data of the decrypted AEAD plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
	resultBuffer	Comment	Contains the verification result.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	

	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.
CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

] (SRS_CryptoStack_00090)

8.5.2.8 **CsmSignatureGenerate**

[SWS_Csm_91058] ⌈

Name	CsmSignatureGenerate	
Comment	Asynchronous processing interface to generate a signature.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob	
Comments	Cancels the job.

Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

SignatureGenerate			
Comments	Operation to generate a signature.		
Variation	--		
Parameters	dataBuffer	Comment	References the data from which the signature shall be generated.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which the signature shall be generated.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	References the signature.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the signature.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	

	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.
	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

] (SRS_CryptoStack_00090)

8.5.2.9 **CsmSignatureVerify**

[SWS_Csm_91059] [

Name	CsmSignatureVerify	
Comment	Asynchronous processing interface to execute the signature verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	9	CSM_E_KEY_NOT_VALID
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

SignatureVerify			
Comments	Operation to verify a signature.		
Variation	--		
Parameters	dataBuffer	Comment	References the data for which signature shall be verified.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for which signature shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
	compareBuffer	Comment	References the signature to be verified.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	compareLength	Comment	Contains the length in bytes of the signature to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the verification result.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_NOT_VALID	Request failed, the key is not valid.	

	CSM_E_KEY_SIZE_MISMATCH	Request failed, a key element has the wrong size.
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.

] (SRS_CryptoStack_00090)

8.5.2.10 CsmRandomGenerate

[SWS_Csm_91060] [

Name	CsmRandomGenerate	
Comment	Asynchronous processing interface to execute the random number generation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	4	CSM_E_ENTROPY_EXHAUSTION
	12	CSM_E_JOB_CANCELED

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Request successful, job has been removed; or job is currently not actively processed.
	E_NOT_OK	Request failed, job couldn't be removed.
	CSM_E_JOB_CANCELED	The job has been marked to be aborted at the next opportunity. It will be further processed until the job finish notification.

RandomGenerate			
Comments	Generates a random number and stores it in the memory location pointed by the resultBuffer pointer.		
Variation	--		
Parameters	resultBuffer	Comment	References the random number.

		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the data of random number.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	Request failed, service is still busy.	
	CSM_E_ENTROPY_EXHAUSTION	Request failed, entropy of random number generator is exhausted.	

] (SRS_CryptoStack_00090)

8.5.3 Client-Server-Interfaces (Key Management)

8.5.3.1 CsmJobKeySetValid

[SWS_Csm_91035] [

Name	CsmJobKeySetValid	
Comment	Interface to set a key valid.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	12	CSM_E_JOB_CANCELED

Operations

CancelJob	
Comments	Cancels the job.
Variation	--

Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_JOB_CANCELED	Failed, service is still busy
KeySetValid		
Comments	Operation to set a key valid.	
Variation	--	
Parameters	key	Comment: Identifier of the key. Type: uint32 Variation: -- Direction: IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_BUSY	Failed, service is still busy

] ()

8.5.3.2 **CsmJobRandomSeed**

[SWS_Csm_91036] [

Name	CsmJobRandomSeed	
Comment	Interface to random seed operation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	12	CSM_E_JOB_CANCELED

Operations

CancelJob	
Comments	Cancels the job.
Variation	--

Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_JOB_CANCELED	Failed, service is still busy	
RandomSeed			
Comments	Utilize the random seed service.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
	seedPtr	Comment	Holds the data which shall be used for the random seed initialization.
		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	seedLength	Comment	Contains the length of the seed in bytes.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	

] ()

8.5.3.3 **CsmJobKeyGenerate**

[SWS_Csm_91037] [

Name	CsmJobKeyGenerate
Comment	Interface to execute key generation.
IsService	true
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}

Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_JOB_CANCELED	Failed, service is still busy

KeyGenerate			
Comments	Generates new key material and stores it in the key identified by keyId.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] ()

8.5.3.4 **CsmJobKeyDerive**

[SWS_Csm_91038] [

Name	CsmJobKeyDerive

Comment	Interface to execute key derive.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	6	CSM_E_KEY_READ_FAIL
	7	CSM_E_KEY_WRITE_FAIL
	10	CSM_E_KEY_SIZE_MISMATCH
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_JOB_CANCELED	Failed, service is still busy

KeyDerive			
Comments	Derives a new key by using the key elements in the given key. 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 targetCryptoKeyId.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyId	Comment	Holds the identifier of the key which is used to store the derived key.
		Type	uint32

		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_SIZE_MISMATCH	Request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] ()

8.5.3.5 **CsmJobKeyExchangeCalcPubVal**

[SWS_Csm_91039] [

Name	CsmJobKeyExchangeCalcPubVal	
Comment	Interface to execute calculation of the public value for key exchange.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob	
Comments	Cancels the job.
Variation	--

Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_JOB_CANCELED	Failed, service is still busy	
KeyExchangeCalcPubVal			
Comments	Calculates the public value of the current user for the key exchange and stores the public key in the provided buffer.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
	publicValuePtr	Comment	Holds a pointer to the memory location in which the public value length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
		Type	Csm_DataPtr
		Variation	--
		Direction	OUT
	publicValueLengthPtr	Comment	Contains the pointer to the data where the public value shall be stored.
		Type	uint32
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] ()

8.5.3.6 CsmJobKeyExchangeCalcSecret

[SWS_Csm_91040] ⌈

Name	CsmJobKeyExchangeCalcSecret	
Comment	Interface to execute calculation of shared secret for key exchange.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_JOB_CANCELED	Failed, service is still busy

KeyExchangeCalcSecret			
Comments	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.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
	partnerPublicValuePtr	Comment	Holds the pointer to the memory location containing the partner's public value.

		Type	Csm_DataPtr
		Variation	--
		Direction	IN
	partnerPublicValueLength	Comment	Contains the number of bytes of the partner public value.
		Type	uint32
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	
	CSM_E_SMALL_BUFFER	The provided buffer is too small to store the result.	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] ()

8.5.3.7 **CsmJobCertificateParse**

[SWS_Csm_91041] [

Name	CsmJobCertificateParse	
Comment	Interface to execute certificate parsing.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob	
Comments	Cancels the job.
Variation	--

Possible Errors	E_OK	Operation successful
	E_NOT_OK	Operation failed
	CSM_E_JOB_CANCELED	Failed, service is still busy

CertificateParse			
Comments	This function shall dispatch the certificate parse function to the CRYIF.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] ()

8.5.3.8 **CsmJobCertificateVerify**

[SWS_Csm_91042] [

Name	CsmJobCertificateVerify	
Comment	Interface to execute certificate verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	12	CSM_E_JOB_CANCELED
	13	CSM_E_KEY_EMPTY

Operations

CancelJob			
Comments	Cancels the job.		
Variation	--		
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_JOB_CANCELED	Failed, service is still busy	

CertificateVerify			
Comments	Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId. Note: Only certificates stored in the same Crypto Driver can be verified against each other. If the key element CRYPTO_KE_CERTIFICATE_CURRENT_TIME is used for the verification of the validity period of the certificate identified by verifyKeyId, it shall have the same format as the timestamp in the certificate.		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	uint32
		Variation	--
		Direction	IN
	verifyKeyId	Comment	Holds the identifier of the key containing the certificate to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	verifyPtr	Comment	Contains the result of the certificate verification.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	Operation failed	
	CSM_E_BUSY	Failed, service is still busy	
	CSM_E_KEY_EMPTY	The service request failed because of uninitialized source key element.	

] ()

8.5.3.9 CallbackNotification

[SWS_Csm_00928] [

Name	CallbackNotification	
Comment	Interface for the callback notification.	
IsService	true	
Variation	--	
Possible Errors	--	--

Operations

CallbackNotification			
Comments	Notifies the application with a return value that the job has finished.		
Variation	--		
Parameters	result	Comment	Return value that shall be returned to the application
		Type	Csm_ResultType
		Variation	--
		Direction	IN

] (SRS_CryptoStack_00090)

8.5.4 Implementation Data Types

8.5.4.1 Crypto_OperationModeType

[SWS_Csm_01029] [

Name	Crypto_OperationModeType		
Kind	Enumeration		
Range	CRYPTO_OPERATIONMODE_START	0x01	Operation Mode is "Start". The job's state shall be reset, i.e. previous input data and intermediate results shall be deleted.
	CRYPTO_OPERATIONMODE_UPDATE	0x02	Operation Mode is "Update". Used to calculate intermediate results.

	CRYPTO_OPERATIONMODE_STREAMSTART	0x03	Operation Mode is "Stream Start". Mixture of "Start" and "Update". Used for streaming.
	CRYPTO_OPERATIONMODE_FINISH	0x04	Operation Mode is "Finish". The calculations shall be finalized.
	CRYPTO_OPERATIONMODE_SINGLECALL	0x07	Operation Mode is "Single Call". Mixture of "Start", "Update" and "Finish".
Description	Enumeration which operation shall be performed. This enumeration is constructed from a bit mask, where the first bit indicates "Start", the second "Update" and the third "Finish".		
Variation	--		
Available via	Rte_Csm_Type.h		

] ()

8.5.4.2 **Crypto_VerifyResultType**

[SWS_Csm_01024] [

Name	Crypto_VerifyResultType		
Kind	Enumeration		
Range	CRYPTO_E_VER_OK	0x00	The result of the verification is "true", i.e. the two compared elements are identical. This return code shall be given as value "0"
	CRYPTO_E_VER_NOT_OK	0x01	The result of the verification is "false", i.e. the two compared elements are not identical. This return code shall be given as value "1".
Description	Enumeration of the result type of verification operations.		
Variation	--		
Available via	<none>		

] ()

8.5.4.3 **Csm_KeyDataType_{Crypto}**

[SWS_Csm_00828] [

Name	Csm_KeyDataType_{Crypto}		
Kind	Array	Element type	uint8

Size	max({ecuc(Csm/CsmKeys/CsmKey/CsmKeyRef->CryIfKey/CryIfKeyRef->CryptoKey/CryptoKeyTypeRef->CryptoKeyType/CryptoKeyElementRef->CryptoKeyElement/CryptoKeyElementSize) Elements
Description	Array long enough to store keys of all types
Variation	Crypto = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}
Available via	Rte_Csm_Type.h

] ()

8.5.4.4 **Csm_ResultType**

[SWS_Csm_91001] [

Name	Csm_ResultType		
Kind	Type		
Derived from	Std_ReturnType		
Description	Csm module specific return values for use in Std_ReturnType that could occur on async.		
Range	E_OK	0x00	The service request is successful.
	E_NOT_OK	0x01	The service request failed.
	E_SMALL_BUFFER	0x02	The service request failed because the provided buffer is too small to store the result.
	E_ENTROPY_EXHAUSTION	0x03	The service request failed because the entropy of random number generator is exhausted.
	E_KEY_READ_FAIL	0x04	The service request failed because read access was denied.
	E_KEY_NOT_AVAILABLE	0x05	The service request failed because the key is not available.
	E_KEY_NOT_VALID	0x06	The service request failed because key was not valid.
	E_JOB_CANCELED	0x07	The service request failed because the job was canceled
	E_KEY_EMPTY	0x08	The service request failed because of uninitialized source key element.
Variation	--		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00095)

8.5.4.5 Csm_HashDataType_{Crypto}

[SWS_Csm_01920] ⌈

Name	Csm_HashDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig/CsmHashDataMaxLength} Elements		
Description	Array long enough to store the data which shall be hashed.		
Variation	Crypto={ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋ (SRS_CryptoStack_00090)

8.5.4.6 Csm_HashResultType_{Crypto}

[SWS_Csm_00912] ⌈

Name	Csm_HashResultType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig/CsmHashResultLength} Elements		
Description	Array long enough to store the data of the hash.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋ (SRS_CryptoStack_00090)

8.5.4.7 Csm_MacGenerateDataType_{Crypto}

[SWS_Csm_00935] ⌈

Name	Csm_MacGenerateDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmMacGenerate/CsmMacGenerateConfig/ CsmMacGenerateDataMaxLength} Elements		
Description	Array long enough to store the data from which a MAC shall be generated.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.8 Csm_MacGenerateResultType_{Crypto}

[SWS_Csm_00927] [

Name	Csm_MacGenerateResultType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmMacGenerate/CsmMacGenerateConfig/ CsmMacGenerateResultLength} Elements		
Description	Array long enough to store the data of the MAC.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.9 Csm_MacVerifyDataType_{Crypto}

[SWS_Csm_00802] [

Name	Csm_MacVerifyDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig/ CsmMacVerifyDataMaxLength} Elements		
Description	Array long enough to store the data for whichs MAC shall be verified.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.10 Csm_MacVerifyCompareType_{Crypto}

[SWS_Csm_00803] [

Name	Csm_MacVerifyCompareType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig/ CsmMacVerifyCompareLength}/8 Elements		
Description	Array long enough to store a MAC to be verified.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		

Available via	Rte_Csm_Type.h

] (SRS_CryptoStack_00090)

8.5.4.11 Csm_EncryptDataType_{Crypto}

[SWS_Csm_01921] [

Name	Csm_EncryptDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmEncrypt/CsmEncryptConfig/ CsmEncryptDataMaxLength} Elements		
Description	Array long enough to store the data to be encrypted.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.12 Csm_EncryptResultType_{Crypto}

[SWS_Csm_01922] [

Name	Csm_EncryptResultType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmEncrypt/CsmEncryptConfig/ CsmEncryptResultMaxLength} Elements		
Description	Array long enough to store the data of the cipher.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.13 Csm_DecryptDataType_{Crypto}

[SWS_Csm_01923] [

Name	Csm_DecryptDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig/ CsmDecryptDataMaxLength} Elements		

Description	Array long enough to store the data to be decrypted.
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}
Available via	Rte_Csm_Type.h

] (SRS_CryptoStack_00090)

8.5.4.14 Csm_DecryptResultType_{Crypto}

[SWS_Csm_01924] [

Name	Csm_DecryptResultType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig/ CsmDecryptResultMaxLength} Elements		
Description	Array long enough to store the data of the decrypted plaintext.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.15 Csm_AEADEncryptPlaintextType_{Crypto}

[SWS_Csm_01925] [

Name	Csm_AEADEncryptPlaintextType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig/ CsmAEADEncryptPlaintextMaxLength} Elements		
Description	Array long enough to store the plaintext to be encrypted with AEAD.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.16 Csm_AEADEncryptAssociatedDataType_{Crypto}

[SWS_Csm_01928] [

Name	Csm_AEADEncryptAssociatedDataType_{Crypto}		
Kind	Array	Element type	uint8

Size	{ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig/ CsmAEADEncryptAssociatedDataMaxLength} Elements
Description	Array long enough to store the data of the header.
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}
Available via	Rte_Csm_Type.h

] (SRS_CryptoStack_00090)

8.5.4.17 Csm_AEADEncryptCiphertextType_{Crypto}

[SWS_Csm_01927] [

Name	Csm_AEADEncryptCiphertextType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig/ CsmAEADEncryptCiphertextMaxLength} Elements		
Description	Array long enough to store the data of the cipher.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.18 Csm_AEADEncryptTagType_{Crypto}

[SWS_Csm_01926] [

Name	Csm_AEADEncryptTagType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig/ CsmAEADEncryptTagLength)} Elements		
Description	Array long enough to store the data of the Tag.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.19 Csm_AEADDecryptCiphertextType_{Crypto}

[SWS_Csm_00922] [

Name	Csm_AEADDecryptCiphertextType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmAEADDecrypt/CsmAEADDecryptConfig/ CsmAEADDecryptCiphertextMaxLength} Elements		
Description	Array long enough to store the ciphertext to be decrypted with AEAD.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.20 Csm_AEADDecryptAssociatedDataType_{Crypto}

[SWS_Csm_00923] [

Name	Csm_AEADDecryptAssociatedDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmAEADDecrypt/CsmAEADDecryptConfig/ CsmAEADDecryptAssociatedDataMaxLength} Elements		
Description	Array long enough to store the data of the header.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.21 Csm_AEADDecryptTagType_{Crypto}

[SWS_Csm_01074] [

Name	Csm_AEADDecryptTagType_{Crypto}		
Kind	Array	Element type	uint8
Size	(({ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig/ CsmAEADDecryptTagLength)}+7)/8) Elements		
Description	Array long enough to store the data of the Tag.		
Variation	Crypto = {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.22 Csm_AEADDecryptPlaintextType_{Crypto}

[SWS_Csm_01075] ⌈

Name	Csm_AEADDecryptPlaintextType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig/ CsmAEADDecryptPlaintextMaxLength} Elements		
Description	Array long enough to store the data of the plaintext.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋ (SRS_CryptoStack_00090)

8.5.4.23 Csm_SignatureGenerateDataType_{Crypto}

[SWS_Csm_01083] ⌈

Name	Csm_SignatureGenerateDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/CsmSignatureGenerateConfig/ CsmSignatureGenerateDataMaxLength} Elements		
Description	Array long enough to store the data from which the signature shall be generated.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋ (SRS_CryptoStack_01076)

8.5.4.24 Csm_SignatureGenerateResultType_{Crypto}

[SWS_Csm_01077] ⌈

Name	Csm_SignatureGenerateResultType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/CsmSignatureGenerateConfig/ CsmSignatureGenerateResultLength} Elements		
Description	Array long enough to store the signature and its length.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.25 Csm_SignatureVerifyDataType_{Crypto}

[SWS_Csm_01078] [

Name	Csm_SignatureVerifyDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmSignatureVerify/CsmSignatureVerifyConfig/ CsmSignatureVerifyDataMaxLength} Elements		
Description	Array long enough to store the data for whichs signature shall be verified.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.26 Csm_SignatureVerifyCompareType_{Crypto}

[SWS_Csm_01079] [

Name	Csm_SignatureVerifyCompareType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmSignatureVerify/CsmSignatureVerifyConfig/ CsmSignatureVerifyCompareLength} Elements		
Description	Array long enough to store a signature to be verified.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

] (SRS_CryptoStack_00090)

8.5.4.27 Csm_RandomGenerateResultType_{Crypto}

[SWS_Csm_00930] [

Name	Csm_RandomGenerateResultType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmRandomGenerate/CsmRandomGenerateConfig/ CsmRandomGenerateResultLength) Elements		
Description	Array long enough to store the data of the random number.		
Variation	Crypto= {ecuc/Csm/CsmPrimitives.SHORT-NAME}		

Available via	Rte_Csm_Type.h

] (SRS_CryptoStack_00090)

8.5.5 Ports

8.5.5.1 CsmKey_{Key}

[SWS_Csm_01042] [

Name	CsmKey_{Key}		
Kind	ProvidedPort	Interface	CsmKeyManagement_{Key}
Description	Port related to a specific cryptographic key to execute the key management functions synchronously.		
Port Defined Argument Value(s)	Type	uint32	
	Value	{ecuc(Csm/CsmKeys/CsmKey/CsmKeyId)}	
Variation	{ecuc(Csm/CsmKeys/CsmKey.CsmKeyUsePort)} == TRUE Key = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.5.5.2 CsmJob_{Job} (CRYPTO_USE_PORT)

[SWS_Csm_91023] [

Name	CsmJob_{Job}		
Kind	ProvidedPort	Interface	{Primitive}_{PrimitiveCfg}
Description	Port related to a specific cryptographic job to execute the assigned cryptographic calculations synchronously.		
Port Defined Argument Value(s)	Type	uint32	
	Value	{ecuc(Csm/CsmJobs/CsmJob.CsmJobId)}	
	Type	Crypto_OperationModeType	
	Value	CRYPTO_OPERATIONMODE_SINGLECALL	
Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmJobInterfaceUsePort)} == CRYPTO_USE_PORT) &&({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef)} != NULL) Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)} Primitive = {ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef->CsmPrimitives/*.SHORT-NAME)} PrimitiveCfg = {ecuc(Csm/CsmPrimitives/{Primitive}/{Primitive}Config.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.5.5.3 CsmJob_{Job} (CRYPTO_USE_PORT_OPTIMIZED)

[SWS_Csm_91062] ⌈

Name	CsmJob_{Job}		
Kind	ProvidedPort	Interface	{Primitive}
Description	Port related to a specific cryptographic job to execute the assigned cryptographic calculations asynchronously.		
Port Defined Argument Value(s)	Type	uint32	
	Value	{ecuc(Csm/CsmJobs/CsmJob.CsmJobId)}	
	Type	Crypto_OperationModeType	
	Value	CRYPTO_OPERATIONMODE_SINGLECALL	
Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmJobInterfaceUsePort)} == CRYPTO_USE_PORT_OPTIMIZED) &&({ecuc(Csm/CsmJobs/CsmJob. CsmJobPrimitiveRef)} != NULL) Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)} Primitive = {ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef->CsmPrimitives/*. SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.5.5.4 {Callback}_CallbackNotification

[SWS_Csm_00934] ⌈

Name	{Job}_CallbackNotification		
Kind	RequiredPort	Interface	CallbackNotification
Description	Port for the callback notification.		
Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmProcessingMode)} == CRYPTO_PROCESSING_ASYNC) Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)}		

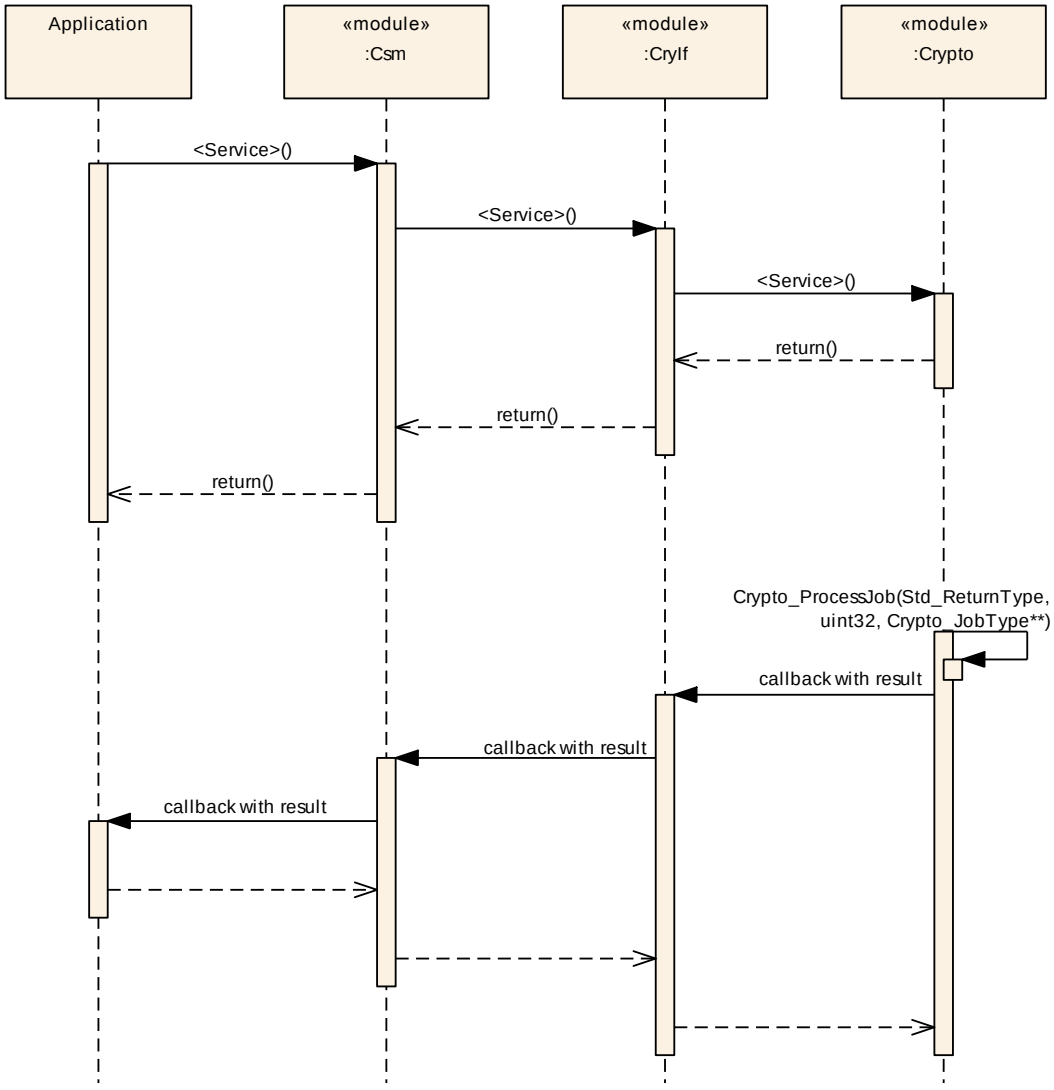
] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

9 Sequence Diagrams

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

9.1.1 Asynchronous Calls

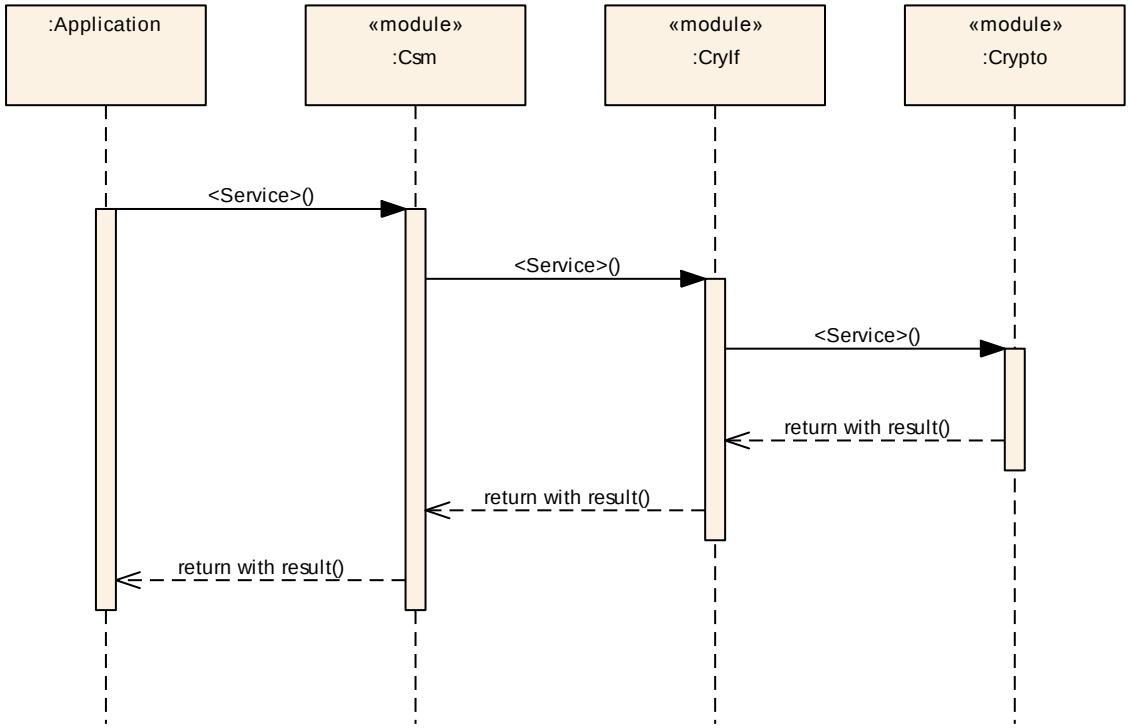
The following diagram (Sequence diagram for asynchronous call) shows a sample sequence of function calls for a request performed asynchronously. The result of the asynchronous function can be accessed after an asynchronous notification (invocation of the configured callback function).



Sequence diagram for asynchronous call with callback

9.1.2 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.



Sequence diagram for synchronous call

10 Configuration

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification.

Chapter 10.2 specifies the structure (containers) and the parameters of the module CSM.

Chapter 10.3 specifies published information of the module CSM.

10.1 How to Read this Chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and Configuration Parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

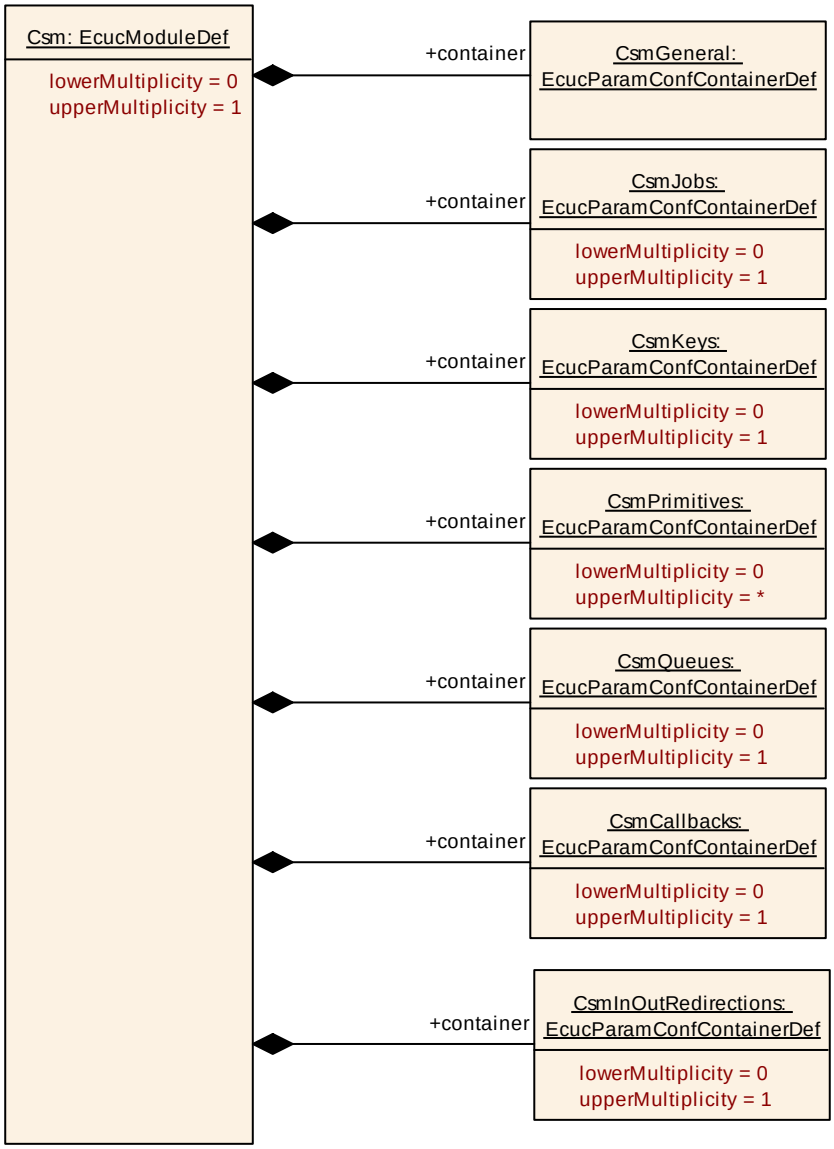


Figure 9-1 Crypto Service Manager Layout

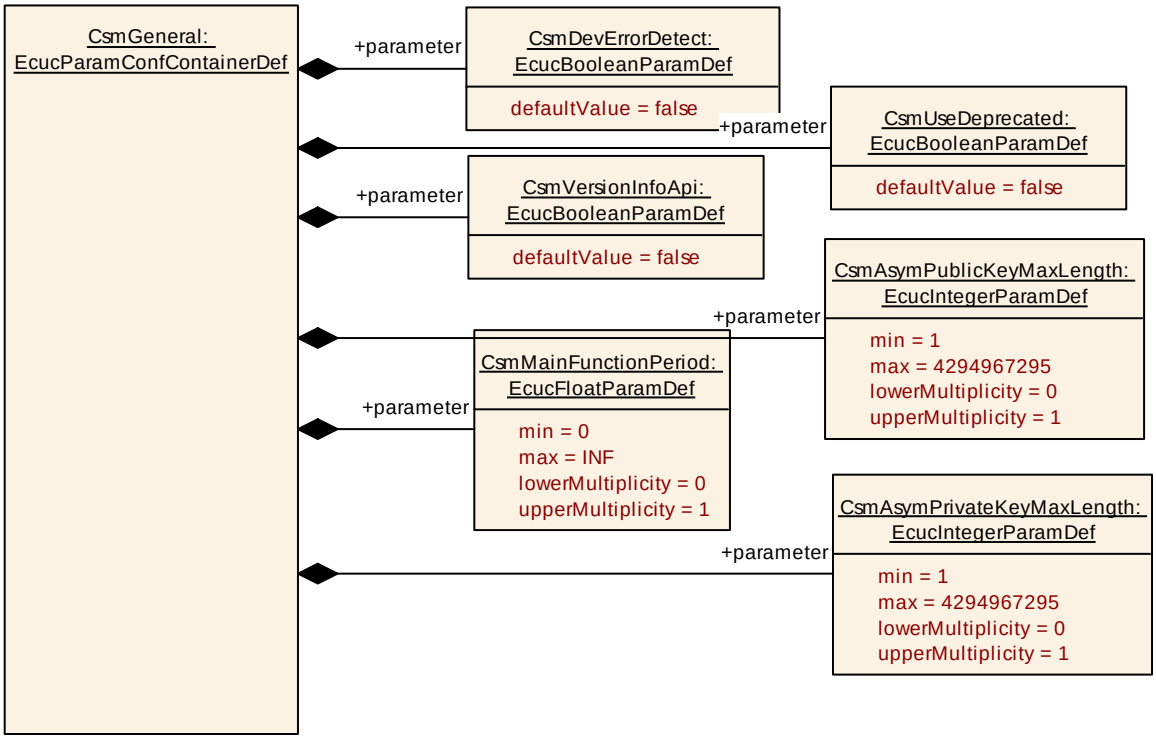


Figure 9-2 Crypto Service Manager General Layout

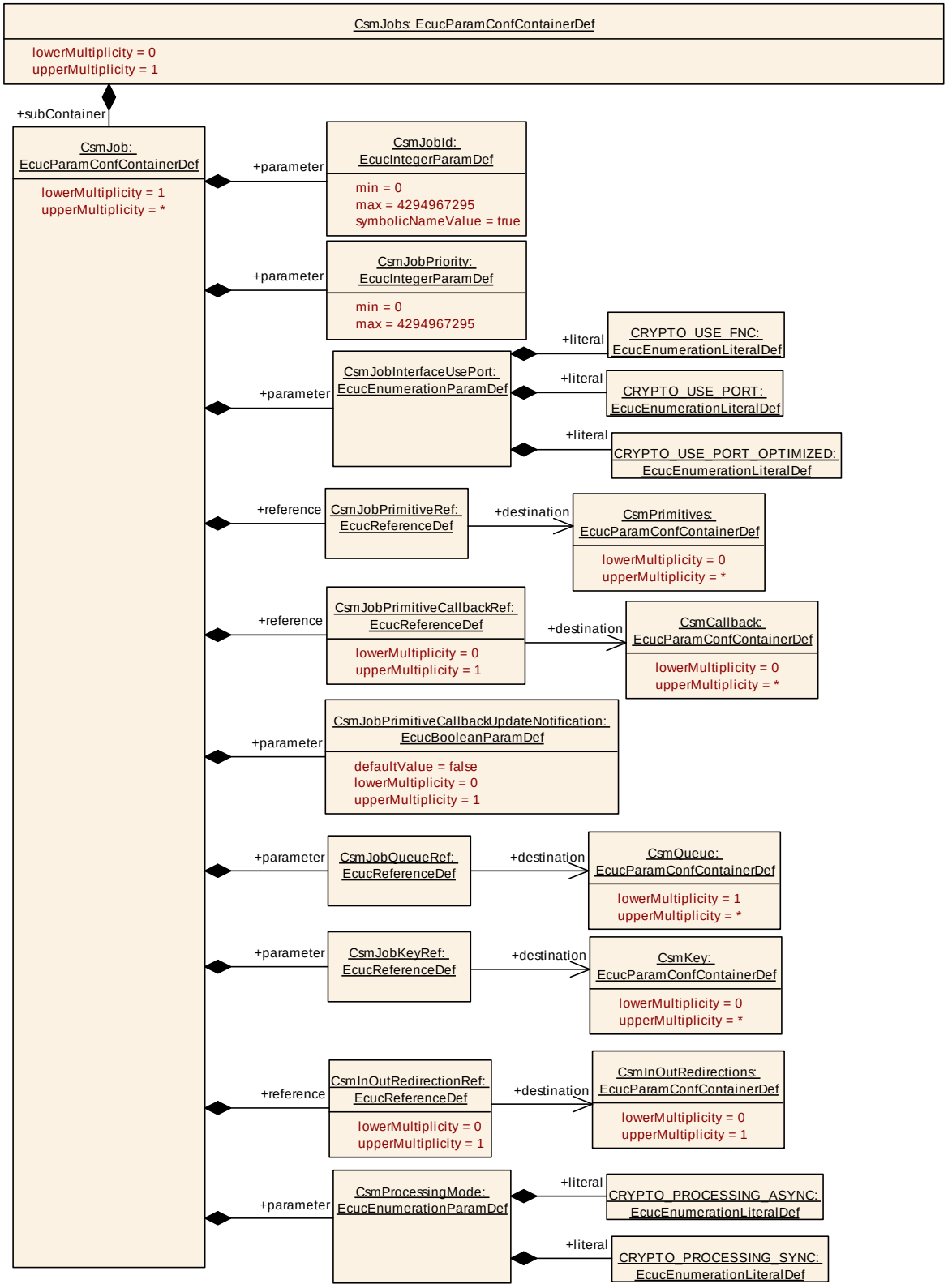


Figure 9-3 Crypto Service Manager Jobs Layout

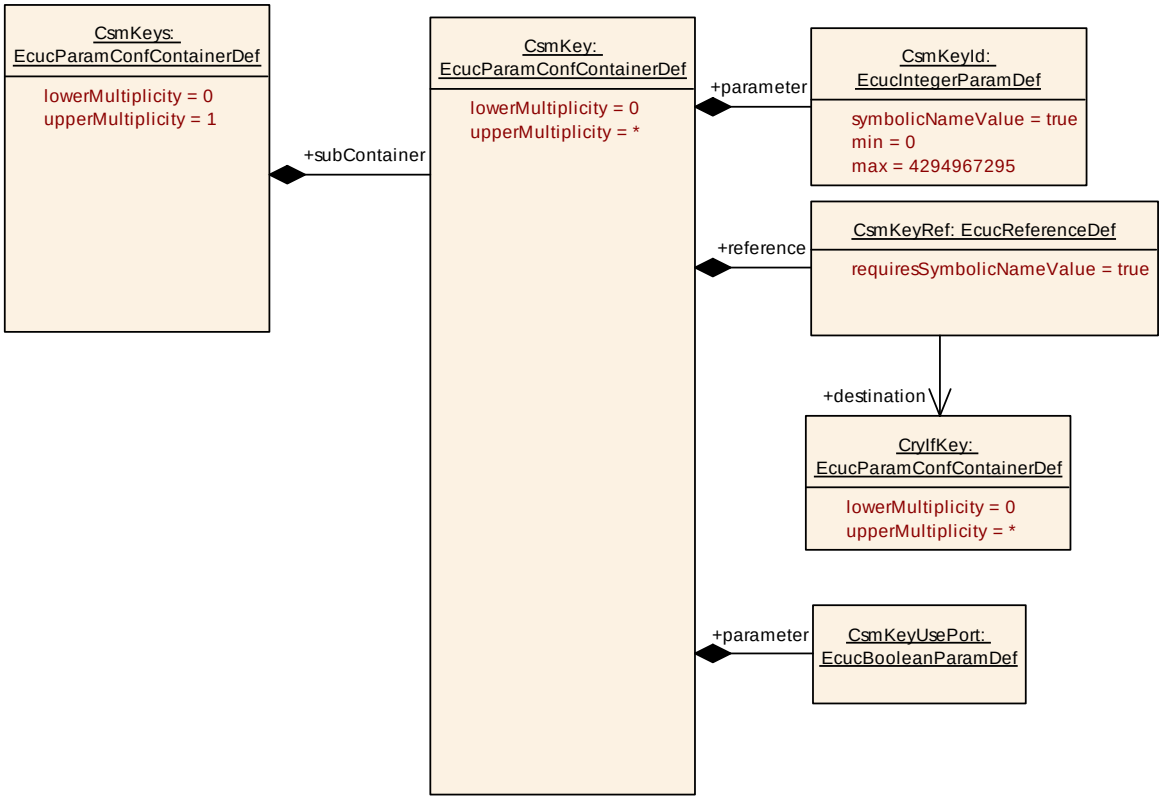


Figure 9-4 Crypto Service Manager Keys Layout

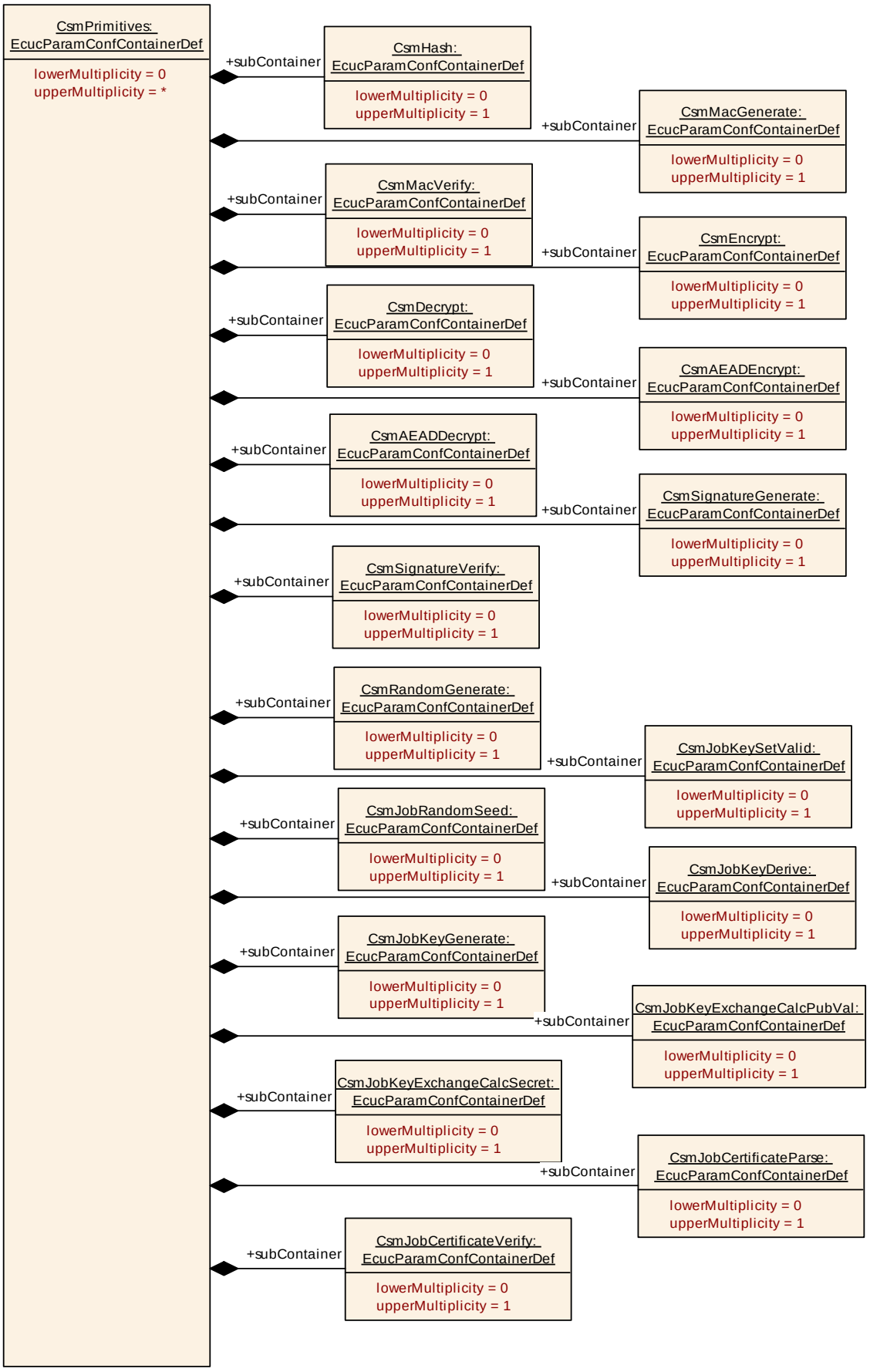


Figure 9-5 Crypto Service Manager Primitives Layout

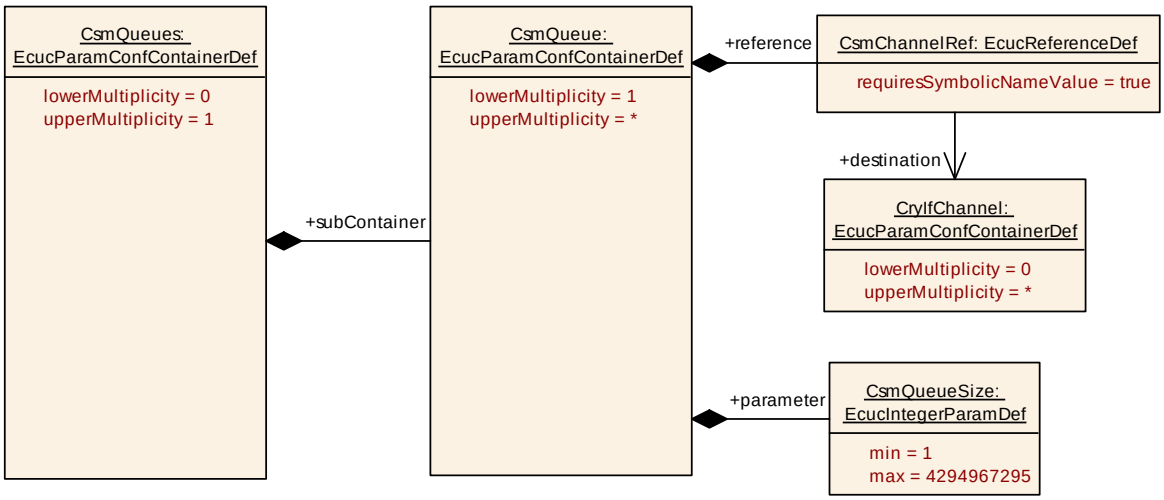


Figure 9-6 Crypto Service Manager Queues Layout

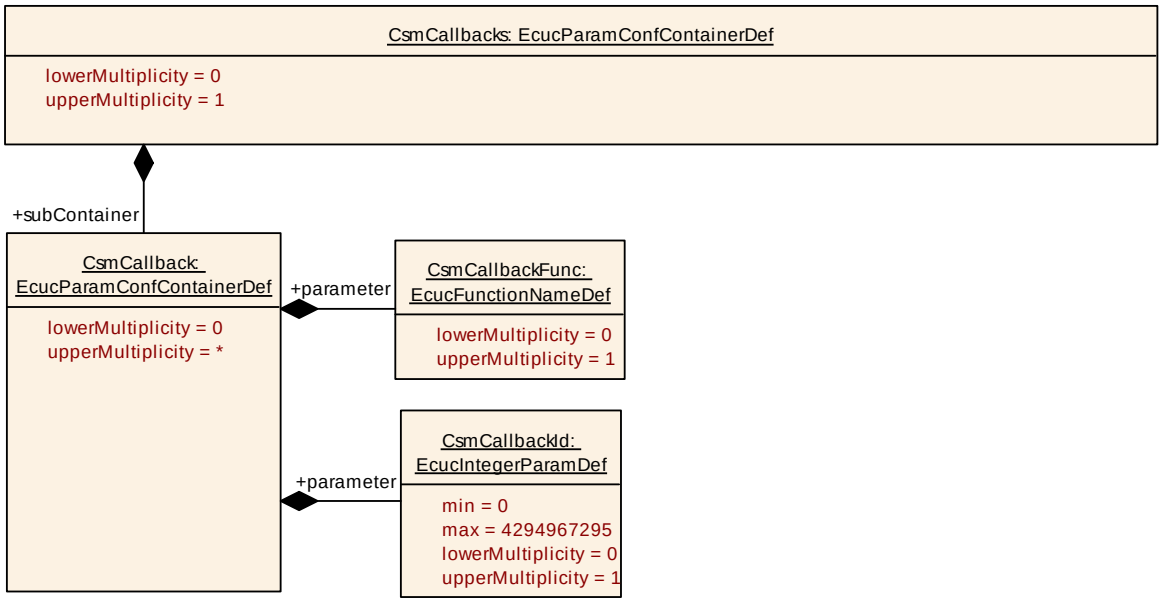


Figure 9-7 Crypto Service Manager Callbacks

10.2.1 **Csm**

SWS Item	**ECUC_Csm_00818 :**
Module Name	*Csm*
Module Description	Configuration of the Csm (CryptoServiceManager) module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmCallbacks	0..1	Container for callback function configurations
CsmGeneral	1	Container for common configuration options.
CsmInOutRedirections	0..1	Configuration for CSM redirection configurations
CsmJobs	0..1	Container for configuration of CSM jobs.

CsmKeys	0..1	Container for CSM key configurations.
CsmPrimitives	0..*	Container for configuration of CsmPrimitives
CsmQueues	0..1	Container for CSM queue configurations

10.2.2 **CsmGeneral**

SWS Item	**ECUC_Csm_00002 :**		
Container Name	CsmGeneral		
Description	Container for common configuration options.		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_Csm_00115 :**		
Name	CsmAsymPrivateKeyMaxLength		
Parent Container	CsmGeneral		
Description	Maximum length in bytes of an asymmetric public key for all algorithm		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00114 :**		
Name	CsmAsymPublicKeyMaxLength		
Parent Container	CsmGeneral		
Description	Maximum length in bytes of an asymmetric key for all algorithm		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00001 :**
Name	CsmDevErrorDetect
Parent Container	CsmGeneral
Description	Switches the development error detection and notification on or off.

	• true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00113 :**		
Name	CsmMainFunctionPeriod		
Parent Container	CsmGeneral		
Description	Specifies the period of main function Csm_MainFunction in seconds.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00117 :**		
Name	CsmUseDeprecated		
Parent Container	CsmGeneral		
Description	Decides if the deprecated interfaces shall be used (Backwards combatibility). true: use deprecated interfaces. false: use normal interfaces.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00003 :**

Name	CsmVersionInfoApi		
Parent Container	CsmGeneral		
Description	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.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.3 CsmJobs

SWS Item	**ECUC_Csm_00112 :**		
Container Name	CsmJobs		
Description	Container for configuration of CSM jobs.		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmJob	1..*	Container for configuration of CSM job. The container name serves as a symbolic name for the identifier of a job configuration.

10.2.4 CsmJob

SWS Item	**ECUC_Csm_00118 :**		
Container Name	CsmJob		
Description	Container for configuration of CSM job. The container name serves as a symbolic name for the identifier of a job configuration.		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_Csm_00119 :**
Name	CsmJobId
Parent Container	CsmJob
Description	Identifier of the CSM job. The set of actually configured identifiers shall be consecutive and gapless.

Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00275 :**		
Name	CsmJobInterfaceUsePort		
Parent Container	CsmJob		
Description	Does the job need RTE interfaces?		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_USE_FNC		Port is not used.
	CRYPTO_USE_PORT		Port is used.
	CRYPTO_USE_PORT_OPTIMIZED		DATA_REFERENCE is used.
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00124 :**		
Name	CsmJobPrimitiveCallbackUpdateNotification		
Parent Container	CsmJob		
Description	This parameter indicates, whether the callback function shall be called, if the UPDATE operation has been finished.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00120 :**		
Name	CsmJobPriority		
Parent Container	CsmJob		
Description	Priority of the job. The higher the value, the higher the job's priority.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		

Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00276 :**		
Name	CsmProcessingMode		
Parent Container	CsmJob		
Description	Determines how the interface shall be used for that job. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_PROCESSING_ASYNC	--	
	CRYPTO_PROCESSING_SYNC	--	
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00263 :**		
Name	CsmInOutRedirectionRef		
Parent Container	CsmJob		
Description	This parameter refers to the used redirection.		
Multiplicity	0..1		
Type	Reference to [CsmInOutRedirections]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00126 :**		
Name	CsmJobKeyRef		
Parent Container	CsmJob		
Description	This parameter refers to the key which shall be used for the CsmPrimitive. It's possible to use a CsmKey for different jobs		
Multiplicity	1		
Type	Reference to [CsmKey]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00123 :**		
Name	CsmJobPrimitiveCallbackRef		
Parent Container	CsmJob		
Description	This parameter refers to the used CsmCallback. The referred CsmCallback is called when the crypto job has been finished.		
Multiplicity	0..1		
Type	Reference to [CsmCallback]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00122 :**		
Name	CsmJobPrimitiveRef		
Parent Container	CsmJob		
Description	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.		
Multiplicity	1		
Type	Reference to [CsmPrimitives]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00125 :**		
Name	CsmJobQueueRef		
Parent Container	CsmJob		
Description	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.		
Multiplicity	1		
Type	Reference to [CsmQueue]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.5 **CsmKeys**

SWS Item	**ECUC_Csm_00005 :**
Container Name	CsmKeys
Description	Container for CSM key configurations.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmKey	0..*	Container for configuration of a CSM key. The container name serves as a symbolic name for the identifier of a key configuration.

10.2.6 **CsmKey**

SWS Item	**ECUC_Csm_00014 :**
Container Name	CsmKey
Description	Container for configuration of a CSM key. The container name serves as a symbolic name for the identifier of a key configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00015 :**		
Name	CsmKeyId		
Parent Container	CsmKey		
Description	Identifier of the CsmKey. The set of actually configured identifiers shall be consecutive and gapless.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00127 :**		
Name	CsmKeyUsePort		
Parent Container	CsmKey		
Description	Does the key need RTE interfaces? True: RTE interfaces used for this key False: No RTE interfaces used for this key		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration	***Pre-compile time***	X	All Variants

Class	***Link time***	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00016 :**		
Name	CsmKeyRef		
Parent Container	CsmKey		
Description	This parameter refers to the used CryIfKey. The underlying CryIfKey refers to a specific CryptoKey in the Crypto Driver.		
Multiplicity	1		
Type	Symbolic name reference to [CryIfKey]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.7 **CsmPrimitives**

SWS Item	**ECUC_Csm_00006 :**
Container Name	CsmPrimitives
Description	Container for configuration of CsmPrimitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAEADDecrypt	0..1	Configuration of AEAD decryption primitives
CsmAEADEncrypt	0..1	Configuration of AEAD encryption primitives
CsmDecrypt	0..1	Configurations of Decryption primitives
CsmEncrypt	0..1	Configurations of Encryption primitives
CsmHash	0..1	Container for Hash Configurations
CsmJobCertificateParse	0..1	Configurations of CertificateParse primitives
CsmJobCertificateVerify	0..1	Configurations of CertificateVerify primitves
CsmJobKeyDerive	0..1	Configurations of KeyDerive primitives
CsmJobKeyExchangeCalcPubVal	0..1	Configurations of KeyExchangeCalcPubVal primitives
CsmJobKeyExchangeCalcSecret	0..1	Configurations of KeyExchangeCalcSecret primitives
CsmJobKeyGenerate	0..1	Configurations of KeyGenerate primitives
CsmJobKeySetValid	0..1	Configurations of KeySetValid primitives
CsmJobRandomSeed	0..1	Configurations of RandomSeed primitives
CsmMacGenerate	0..1	Configurations of MacGenerate primitives
CsmMacVerify	0..1	Configurations of MacVerify primitives
CsmRandomGenerate	0..1	Configurations of RandomGenerate primitives
CsmSignatureGenerate	0..1	Configurations of SignatureGenerate primitives
CsmSignatureVerify	0..1	Configurations of SignatureVerify primitives

10.2.8 **CsmQueues**

SWS Item	**ECUC_Csm_00007 :**
Container Name	CsmQueues
Description	Container for CSM queue configurations
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmQueue	1..*	Container for configuration of a CSM queue. A queue has two tasks: 1. queue jobs which cannot be processed since the underlying hardware is busy and 2. refer to channel which shall be used

10.2.9 **CsmQueue**

SWS Item	**ECUC_Csm_00032 :**
Container Name	CsmQueue
Description	Container for configuration of a CSM queue. A queue has two tasks: 1. queue jobs which cannot be processed since the underlying hardware is busy and 2. refer to channel which shall be used
Configuration Parameters	

SWS Item	**ECUC_Csm_00034 :**		
Name	CsmQueueSize		
Parent Container	CsmQueue		
Description	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.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00033 :**		
Name	CsmChannelRef		
Parent Container	CsmQueue		
Description	Refers to the underlying Crypto Interface channel.		
Multiplicity	1		
Type	Symbolic name reference to [CryIfChannel]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.10 CsmInOutRedirections

SWS Item	**ECUC_Csm_00262 :**
Container Name	CsmInOutRedirections
Description	Configuration for CSM redirection configurations
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmInOutRedirection	1..*	Container for configuration of a CSM redirection. A redirection let a CSM job use a specific key element as input or/and output.

10.2.11 CsmInOutRedirection

SWS Item	**ECUC_Csm_00264 :**
Container Name	CsmInOutRedirection
Description	Container for configuration of a CSM redirection. A redirection let a CSM job use a specific key element as input or/and output.
Configuration Parameters	

SWS Item	**ECUC_Csm_00266 :**		
Name	CsmInputKeyElementId		
Parent Container	CsmInOutRedirection		
Description	Identifier of the key element used as input		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00272 :**
Name	CsmOutputKeyElementId

Parent Container	CsmInOutRedirection		
Description	Identifier of the key element used as output.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00269 :**		
Name	CsmSecondaryInputKeyElementId		
Parent Container	CsmInOutRedirection		
Description	Identifier of the key element used as secondary input.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00274 :**		
Name	CsmSecondaryOutputKeyElementId		
Parent Container	CsmInOutRedirection		
Description	Identifier of the key element used as secondary output.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00270 :**

Name	CsmTertiaryInputKeyElementId		
Parent Container	CsmInOutRedirection		
Description	Identifier of the key element used as tertiary input.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00265 :**		
Name	CsmInputKeyRef		
Parent Container	CsmInOutRedirection		
Description	This parameter refers to the key used as input.		
Multiplicity	0..1		
Type	Reference to [CsmKey]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00271 :**		
Name	CsmOutputKeyRef		
Parent Container	CsmInOutRedirection		
Description	This parameter refers to the key used as output.		
Multiplicity	0..1		
Type	Reference to [CsmKey]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00267 :**
Name	CsmSecondaryInputKeyRef
Parent Container	CsmInOutRedirection
Description	This parameter refers to the key used as secondary input.

Multiplicity	0..1		
Type	Reference to [CsmKey]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00273 :**		
Name	CsmSecondaryOutputKeyRef		
Parent Container	CsmInOutRedirection		
Description	This parameter refers to the key used as secondary output.		
Multiplicity	0..1		
Type	Reference to [CsmKey]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00268 :**		
Name	CsmTertiaryInputKeyRef		
Parent Container	CsmInOutRedirection		
Description	This parameter refers to the key used as tertiary input.		
Multiplicity	0..1		
Type	Reference to [CsmKey]		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.12 CsmHash

SWS Item	**ECUC_Csm_00021 :**
Container Name	CsmHash

Description	Container for Hash Configurations
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmHashConfig	1	Container for configuration of a CSM hash. The container name serves as a symbolic name for the identifier of a key configuration.

10.2.13 CsmHashConfig

SWS Item	**ECUC_Csm_00036 :**
Container Name	CsmHashConfig
Description	Container for configuration of a CSM hash. The container name serves as a symbolic name for the identifier of a key configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00038 :**		
Name	CsmHashAlgorithmFamily		
Parent Container	CsmHashConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00128 :		
Name	CsmHashAlgorithmFamilyCustom		
Parent Container	CsmHashConfig		
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmHashAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00131 :		
Name	CsmHashAlgorithmMode		
Parent Container	CsmHashConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
Default value	CRYPTO_ALGOMODE_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00132 :		
Name	CsmHashAlgorithmModeCustom		
Parent Container	CsmHashConfig		
Description	Name of the custom primitive mode.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00181 :**		
Name	CsmHashAlgorithmSecondaryFamily		
Parent Container	CsmHashConfig		
Description	Determines the algorithm family used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x00	
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00129 :**		
Name	CsmHashAlgorithmSecondaryFamilyCustom		
Parent Container	CsmHashConfig		
Description	This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmHashAlgorithmSecondaryFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00040 :**		
Name	CsmHashDataMaxLength		
Parent Container	CsmHashConfig		
Description	Max size of the input data length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00130 :**		
Name	CsmHashResultLength		
Parent Container	CsmHashConfig		
Description	Size of the output hash length in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.14 CsmMacGenerate

SWS Item	**ECUC_Csm_00022 :**
Container Name	CsmMacGenerate
Description	Configurations of MacGenerate primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmMacGenerateConfig	1	Container for configuration of a CSM mac generation interface. The container name serves as a symbolic name for the identifier of a MAC generation interface.

10.2.15 CsmMacGenerateConfig

SWS Item	**ECUC_Csm_00041 :**
Container Name	CsmMacGenerateConfig
Description	Container for configuration of a CSM mac generation interface. The container name serves as a symbolic name for the identifier of a MAC generation interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00188 :**
Name	CsmMacGenerateAlgorithmFamily
Parent Container	CsmMacGenerateConfig
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.
Multiplicity	1

Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_RNG	0x1B	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
	CRYPTO_ALGOFAM_SIPHASH	0x1C	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00133 :**		
Name	CsmMacGenerateAlgorithmFamilyCustom		
Parent Container	CsmMacGenerateConfig		
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmMacGenerateAlgorithmFamily		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00044 :		
Name	CsmMacGenerateAlgorithmKeyLength		
Parent Container	CsmMacGenerateConfig		
Description	Size of the MAC key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00189 :		
Name	CsmMacGenerateAlgorithmMode		
Parent Container	CsmMacGenerateConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CMAC	0x10	
	CRYPTO_ALGOMODE_CTRDRBG	0x12	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_GMAC	0x11	
	CRYPTO_ALGOMODE_HMAC	0x0f	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
	CRYPTO_ALGOMODE_SIPHASH_2_4	0x17	
	CRYPTO_ALGOMODE_SIPHASH_4_8	0x18	
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00136 :		
Name	CsmMacGenerateAlgorithmModeCustom		
Parent Container	CsmMacGenerateConfig		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	

	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00134 :**		
Name	CsmMacGenerateAlgorithmSecondaryFamily		
Parent Container	CsmMacGenerateConfig		
Description	Determines the secondary algorithm family used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_NOT_SET	0x00	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00135 :**		
Name	CsmMacGenerateAlgorithmSecondaryFamilyCustom		
Parent Container	CsmMacGenerateConfig		
Description	This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmHashAlgorithmSecondaryFamilyCustom.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00137 :**	
Name	CsmMacGenerateDataMaxLength	
Parent Container	CsmMacGenerateConfig	
Description	Max size of the input data length in bytes	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	--	
Post-Build Variant Value	false	

Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00138 :**		
Name	CsmMacGenerateResultLength		
Parent Container	CsmMacGenerateConfig		
Description	Size of the output MAC length in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.16 CsmMacVerify

SWS Item	**ECUC_Csm_00023 :**
Container Name	CsmMacVerify
Description	Configurations of MacVerify primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmMacVerifyConfig	1	Container for configuration of a CSM MAC verification interface. The container name serves as a symbolic name for the identifier of a MAC generation interface

10.2.17 CsmMacVerifyConfig

SWS Item	**ECUC_Csm_00049 :**
Container Name	CsmMacVerifyConfig
Description	Container for configuration of a CSM MAC verification interface. The container name serves as a symbolic name for the identifier of a MAC generation interface
Configuration Parameters	

SWS Item	**ECUC_Csm_00051 :**
Name	CsmMacVerifyAlgorithmFamily
Parent Container	CsmMacVerifyConfig

Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_RNG	0x1B	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
	CRYPTO_ALGOFAM_SIPHASH	0x1C	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00139 :**		
Name	CsmMacVerifyAlgorithmFamilyCustom		
Parent Container	CsmMacVerifyConfig		
Description	Name of the custom algorithm family used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_Csm_00193 :**		
Name	CsmMacVerifyAlgorithmKeyLength		
Parent Container	CsmMacVerifyConfig		
Description	Size of the MAC key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00195 :**		
Name	CsmMacVerifyAlgorithmMode		
Parent Container	CsmMacVerifyConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CMAC	0x10	
	CRYPTO_ALGOMODE_CTRDRBG	0x12	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_GMAC	0x11	
	CRYPTO_ALGOMODE_HMAC	0x0f	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
	CRYPTO_ALGOMODE_SIPHASH_2_4	0x17	
	CRYPTO_ALGOMODE_SIPHASH_4_8	0x18	
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00194 :**		
Name	CsmMacVerifyAlgorithmModeCustom		
Parent Container	CsmMacVerifyConfig		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00140 :**		
Name	CsmMacVerifyAlgorithmSecondaryFamily		
Parent Container	CsmMacVerifyConfig		
Description	Determines the secondary algorithm family used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x0f	
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00141 :**		
Name	CsmMacVerifyAlgorithmSecondaryFamilyCustom		
Parent Container	CsmMacVerifyConfig		
Description	This is the second the name of the custom algorithm, if CRYPTO_ALGOFAM_CUSTOM is set as CsmMacVerifyAlgorithmSecondaryFamily		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00142 :**		
Name	CsmMacVerifyCompareLength		
Parent Container	CsmMacVerifyConfig		
Description	Size of the input MAC length, that shall be verified, in BITS		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00056 :**		
Name	CsmMacVerifyDataMaxLength		
Parent Container	CsmMacVerifyConfig		
Description	Max size of the input data length, for whichs MAC shall be verified, in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.18 CsmEncrypt

SWS Item	**ECUC_Csm_00024 :**
Container Name	CsmEncrypt
Description	Configurations of Encryption primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmEncryptConfig	1	Container for configuration of a CSM encryption interface. The container name serves as a symbolic name for the identifier of an encryption interface.

10.2.19 CsmEncryptConfig

SWS Item	**ECUC_Csm_00057 :**
Container Name	CsmEncryptConfig
Description	Container for configuration of a CSM encryption interface. The container name serves as a symbolic name for the identifier of an encryption interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00182 :**
Name	CsmEncryptAlgorithmFamily
Parent Container	CsmEncryptConfig
Description	Determines the algorithm family used for the crypto service. This parameter defines

	the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_ECIES	0x1D	
	CRYPTO_ALGOFAM_RSA	0x16	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00143 :**		
Name	CsmEncryptAlgorithmFamilyCustom		
Parent Container	CsmEncryptConfig		
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmEncryptAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00191 :**		
Name	CsmEncryptAlgorithmKeyLength		
Parent Container	CsmEncryptConfig		
Description	Size of the encryption key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00060 :**		
Name	CsmEncryptAlgorithmMode		
Parent Container	CsmEncryptConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_12ROUNDS	0x0d	
	CRYPTO_ALGOMODE_20ROUNDS	0x0e	
	CRYPTO_ALGOMODE_8ROUNDS	0x0c	
	CRYPTO_ALGOMODE_CBC	0x02	
	CRYPTO_ALGOMODE_CFB	0x03	
	CRYPTO_ALGOMODE_CTR	0x05	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_ECB	0x01	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
	CRYPTO_ALGOMODE_OFB	0x04	
	CRYPTO_ALGOMODE_RSAES_OAEP	0x08	
	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5	0x09	
	CRYPTO_ALGOMODE_XTS	0x06	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00153 :**		
Name	CsmEncryptAlgorithmModeCustom		
Parent Container	CsmEncryptConfig		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00144 :**
Name	CsmEncryptAlgorithmSecondaryFamily
Parent Container	CsmEncryptConfig
Description	Determines the algorithm family used for the crypto service
Multiplicity	1
Type	EcucEnumerationParamDef

Range	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x00	
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00190 :**		
Name	CsmEncryptAlgorithmSecondaryFamilyCustom		
Parent Container	CsmEncryptConfig		
Description	Name of the custom secondary algorithm family used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00146 :**		
Name	CsmEncryptDataMaxLength		
Parent Container	CsmEncryptConfig		
Description	Max size of the input plaintext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00147 :**
Name	CsmEncryptResultMaxLength
Parent Container	CsmEncryptConfig
Description	Max size of the output cipher length in bytes
Multiplicity	0..1
Type	EcucIntegerParamDef

Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.20 CsmDecrypt

SWS Item	**ECUC_Csm_00025 :**
Container Name	CsmDecrypt
Description	Configurations of Decryption primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmDecryptConfig	1	Container for configuration of a CSM decryption interface. The container name serves as a symbolic name for the identifier of an decryption interface.

10.2.21 CsmDecryptConfig

SWS Item	**ECUC_Csm_00064 :**
Container Name	CsmDecryptConfig
Description	Container for configuration of a CSM decryption interface. The container name serves as a symbolic name for the identifier of an decryption interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00066 :**		
Name	CsmDecryptAlgorithmFamily		
Parent Container	CsmDecryptConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_ECIES	0x1D	
	CRYPTO_ALGOFAM_RSA	0x16	
Post-Build Variant Value	false		
Multiplicity Configuration	***Pre-compile time***	X	All Variants
	Link time	--	

Class	***Post-build time***	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00148 :**		
Name	CsmDecryptAlgorithmFamilyCustom		
Parent Container	CsmDecryptConfig		
Description	Name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmDecryptAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00067 :**		
Name	CsmDecryptAlgorithmKeyLength		
Parent Container	CsmDecryptConfig		
Description	Size of the encryption key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00068 :**	
Name	CsmDecryptAlgorithmMode	
Parent Container	CsmDecryptConfig	
Description	Determines the algorithm mode used for the crypto service	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	CRYPTO_ALGOMODE_12ROUNDS	0x0d
	CRYPTO_ALGOMODE_20ROUNDS	0x0e
	CRYPTO_ALGOMODE_8ROUNDS	0x0c
	CRYPTO_ALGOMODE_CBC	0x02
	CRYPTO_ALGOMODE_CFB	0x03
	CRYPTO_ALGOMODE_CTR	0x05

	CRYPTO_ALGOMODE_CUSTOM		0xFF
	CRYPTO_ALGOMODE_ECB		0x01
	CRYPTO_ALGOMODE_OFB		0x04
	CRYPTO_ALGOMODE_RSAES_OAEP		0x08
	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5		0x09
	CRYPTO_ALGOMODE_XTS		0x06
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00152 :**		
Name	CsmDecryptAlgorithmModeCustom		
Parent Container	CsmDecryptConfig		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00149 :**		
Name	CsmDecryptAlgorithmSecondaryFamily		
Parent Container	CsmDecryptConfig		
Description	Determines the secondary algorithm family used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_CUSTOM		0xFF
	CRYPTO_ALGOFAM_NOT_SET		0x00
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00150 :**		
Name	CsmDecryptAlgorithmSecondaryFamilyCustom		
Parent Container	CsmDecryptConfig		
Description	Name of the custom secondary algorithm family used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00154 :**		
Name	CsmDecryptDataMaxLength		
Parent Container	CsmDecryptConfig		
Description	Max size of the input ciphertext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00155 :**		
Name	CsmDecryptResultMaxLength		
Parent Container	CsmDecryptConfig		
Description	Max size of the output plaintext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.22 CsmAEADEncrypt

SWS Item	**ECUC_Csm_00026 :**
Container Name	CsmAEADEncrypt
Description	Configuration of AEAD encryption primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAEADEncryptConfig	1	Container for configuration of a CSM encryption interface. The container name serves as a symbolic name for the identifier of an encryption interface.

10.2.23 CsmAEADEncryptConfig

SWS Item	**ECUC_Csm_00072 :**
Container Name	CsmAEADEncryptConfig
Description	Container for configuration of a CSM encryption interface. The container name serves as a symbolic name for the identifier of an encryption interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00074 :**		
Name	CsmAEADEncryptAlgorithmFamily		
Parent Container	CsmAEADEncryptConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00184 :**
Name	CsmAEADEncryptAlgorithmFamilyCustom
Parent Container	CsmAEADEncryptConfig
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADEncryptAlgorithmFamily.
Multiplicity	0..1
Type	EcucStringParamDef

Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00075 :**		
Name	CsmAEADEncryptAlgorithmKeyLength		
Parent Container	CsmAEADEncryptConfig		
Description	Size of the AEAD encryption key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00076 :**		
Name	CsmAEADEncryptAlgorithmMode		
Parent Container	CsmAEADEncryptConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_GCM	0x07	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00187 :**
Name	CsmAEADEncryptAlgorithmModeCustom
Parent Container	CsmAEADEncryptConfig
Description	Name of the custom algorithm mode used for the crypto service
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--

maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00159 :**		
Name	CsmAEADEncryptAssociatedDataMaxLength		
Parent Container	CsmAEADEncryptConfig		
Description	Max size of the input associated data length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00160 :**		
Name	CsmAEADEncryptCiphertextMaxLength		
Parent Container	CsmAEADEncryptConfig		
Description	Max size of the output ciphertext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00158 :**		
Name	CsmAEADEncryptPlaintextMaxLength		
Parent Container	CsmAEADEncryptConfig		
Description	Max size of the input plaintext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		

Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00161 :**		
Name	CsmAEADEncryptTagLength		
Parent Container	CsmAEADEncryptConfig		
Description	Size of the output Tag length in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00157 :**		
Name	CsmAEADEncryptKeyRef		
Parent Container	CsmAEADEncryptConfig		
Description	This parameter refers to the key used for that encryption primitive.		
Multiplicity	1		
Type	Reference to [CsmKey]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00156 :**		
Name	CsmAEADEncryptQueueRef		
Parent Container	CsmAEADEncryptConfig		
Description	This parameter refers to the queue used for that encryption primitive.		
Multiplicity	1		
Type	Reference to [CsmQueue]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.24 CsmAEADDecrypt

SWS Item	**ECUC_Csm_00027 :**
Container Name	CsmAEADDecrypt
Description	Configuration of AEAD decryption primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAEADDecryptConfig	1	Container for configuration of a CSM decryption interface. The container name serves as a symbolic name for the identifier of an decryption interface.

10.2.25 CsmAEADDecryptConfig

SWS Item	**ECUC_Csm_00080 :**
Container Name	CsmAEADDecryptConfig
Description	Container for configuration of a CSM decryption interface. The container name serves as a symbolic name for the identifier of an decryption interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00082 :**		
Name	CsmAEADDecryptAlgorithmFamily		
Parent Container	CsmAEADDecryptConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00185 :**
Name	CsmAEADDecryptAlgorithmFamilyCustom
Parent Container	CsmAEADDecryptConfig
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADDecryptAlgorithmFamily.

Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00083 :**		
Name	CsmAEADDecryptAlgorithmKeyLength		
Parent Container	CsmAEADDecryptConfig		
Description	Size of the AEAD decryption key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00084 :**		
Name	CsmAEADDecryptAlgorithmMode		
Parent Container	CsmAEADDecryptConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_GCM	0x07	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00186 :**
Name	CsmAEADDecryptAlgorithmModeCustom
Parent Container	CsmAEADDecryptConfig
Description	Name of the custom algorithm mode used for the crypto service
Multiplicity	0..1

Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00163 :**		
Name	CsmAEADDecryptAssociatedDataMaxLength		
Parent Container	CsmAEADDecryptConfig		
Description	Max size of the input associated data length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00162 :**		
Name	CsmAEADDecryptCiphertextMaxLength		
Parent Container	CsmAEADDecryptConfig		
Description	Max size of the input ciphertext in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00165 :**		
Name	CsmAEADDecryptPlaintextMaxLength		
Parent Container	CsmAEADDecryptConfig		
Description	Size of the output plaintext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		

Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00164 :**		
Name	CsmAEADDecryptTagLength		
Parent Container	CsmAEADDecryptConfig		
Description	Size of the input Tag length in BITS		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00086 :**		
Name	CsmAEADDecryptKeyRef		
Parent Container	CsmAEADDecryptConfig		
Description	This parameter refers to the key used for that decryption primitive.		
Multiplicity	1		
Type	Reference to [CsmKey]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00081 :**		
Name	CsmAEADDecryptQueueRef		
Parent Container	CsmAEADDecryptConfig		
Description	This parameter refers to the queue used for that decryption primitive.		
Multiplicity	1		
Type	Reference to [CsmQueue]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

No Included Containers

10.2.26 CsmSignatureGenerate

SWS Item	**ECUC_Csm_00028 :**
Container Name	CsmSignatureGenerate
Description	Configurations of SignatureGenerate primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSignatureGenerateConfig	1	Container for configuration of a CSM signature generation interface. The container name serves as a symbolic name for the identifier of signature generation interface.

10.2.27 CsmSignatureGenerateConfig

SWS Item	**ECUC_Csm_00087 :**
Container Name	CsmSignatureGenerateConfig
Description	Container for configuration of a CSM signature generation interface. The container name serves as a symbolic name for the identifier of signature generation interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00089 :**		
Name	CsmSignatureGenerateAlgorithmFamily		
Parent Container	CsmSignatureGenerateConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_BRAINPOOL	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_ECCNIST	0x16	
	CRYPTO_ALGOFAM_ED25519	0x14	
	CRYPTO_ALGOFAM_RSA	0x13	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00166 :**
Name	CsmSignatureGenerateAlgorithmFamilyCustom

Parent Container	CsmSignatureGenerateConfig		
Description	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.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00091 :**		
Name	CsmSignatureGenerateAlgorithmMode		
Parent Container	CsmSignatureGenerateConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5	0x0b	
	CRYPTO_ALGOMODE_RSASSA_PSS	0x0a	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00168 :**		
Name	CsmSignatureGenerateAlgorithmModeCustom		
Parent Container	CsmSignatureGenerateConfig		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00183 :**		
Name	CsmSignatureGenerateAlgorithmSecondaryFamily		
Parent Container	CsmSignatureGenerateConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x00	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00167 :**
Name	CsmSignatureGenerateAlgorithmSecondaryFamilyCustom
Parent Container	CsmSignatureGenerateConfig
Description	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.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--
Post-Build Variant Value	false

Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00169 :**		
Name	CsmSignatureGenerateDataMaxLength		
Parent Container	CsmSignatureGenerateConfig		
Description	Size of the input data length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00090 :**		
Name	CsmSignatureGenerateKeyLength		
Parent Container	CsmSignatureGenerateConfig		
Description	Size of the signature generate key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00170 :**		
Name	CsmSignatureGenerateResultLength		
Parent Container	CsmSignatureGenerateConfig		
Description	Size of the output signature length in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants

	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.28 CsmSignatureVerify

SWS Item	**ECUC_Csm_00029 :**
Container Name	CsmSignatureVerify
Description	Configurations of SignatureVerify primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSignatureVerifyConfig	1	Container for configuration of a CSM signature verification interface. The container name serves as a symbolic name for the identifier of signature verification interface.

10.2.29 CsmSignatureVerifyConfig

SWS Item	**ECUC_Csm_00094 :**
Container Name	CsmSignatureVerifyConfig
Description	Container for configuration of a CSM signature verification interface. The container name serves as a symbolic name for the identifier of signature verification interface.
Configuration Parameters	

SWS Item	**ECUC_Csm_00096 :**		
Name	CsmSignatureVerifyAlgorithmFamily		
Parent Container	CsmSignatureVerifyConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_BRAINPOOL	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_ECCNIST	0x16	
	CRYPTO_ALGOFAM_ED25519	0x14	
	CRYPTO_ALGOFAM_RSA	0x13	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00171 :**

Name	CsmSignatureVerifyAlgorithmFamilyCustom		
Parent Container	CsmSignatureVerifyConfig		
Description	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.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00098 :**		
Name	CsmSignatureVerifyAlgorithmMode		
Parent Container	CsmSignatureVerifyConfig		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5	0x0B	
	CRYPTO_ALGOMODE_RSASSA_PSS	0x0A	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00174 :**		
Name	CsmSignatureVerifyAlgorithmModeCustom		
Parent Container	CsmSignatureVerifyConfig		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00172 :**		
Name	CsmSignatureVerifyAlgorithmSecondaryFamily		
Parent Container	CsmSignatureVerifyConfig		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x00	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00173 :**
Name	CsmSignatureVerifyAlgorithmSecondaryFamilyCustom
Parent Container	CsmSignatureVerifyConfig
Description	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.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--

Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00176 :**		
Name	CsmSignatureVerifyCompareLength		
Parent Container	CsmSignatureVerifyConfig		
Description	Number of the least significant bytes of the signature, for which the verification shall be calculated.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00175 :**		
Name	CsmSignatureVerifyDataMaxLength		
Parent Container	CsmSignatureVerifyConfig		
Description	Max size of the input data, for which the signature shall be verified, in bytes.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00192 :**		
Name	CsmSignatureVerifyKeyLength		
Parent Container	CsmSignatureVerifyConfig		
Description	Size of the signature verify key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.30 CsmRandomGenerate

SWS Item	**ECUC_Csm_00031 :**
Container Name	CsmRandomGenerate
Description	Configurations of RandomGenerate primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmRandomGenerateConfig	1	Container for configuration of a CSM random generator. The container name serves as a symbolic name for the identifier of a random generator configuration.

10.2.31 CsmRandomGenerateConfig

SWS Item	**ECUC_Csm_00103 :**
Container Name	CsmRandomGenerateConfig
Description	Container for configuration of a CSM random generator. The container name serves as a symbolic name for the identifier of a random generator configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00105 :**	
Name	CsmRandomGenerateAlgorithmFamily	
Parent Container	CsmRandomGenerateConfig	
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	CRYPTO_ALGOFAM_3DES	0x13
	CRYPTO_ALGOFAM_AES	0x14
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12
	CRYPTO_ALGOFAM_CHACHA	0x15
	CRYPTO_ALGOFAM_CUSTOM	0xFF
	CRYPTO_ALGOFAM_RIPEMD160	0x0E
	CRYPTO_ALGOFAM_RNG	0x1B
	CRYPTO_ALGOFAM_SHA1	0x01
	CRYPTO_ALGOFAM_SHA2_224	0x02
	CRYPTO_ALGOFAM_SHA2_256	0x03

	CRYPTO_ALGOFAM_SHA2_384		0x04
	CRYPTO_ALGOFAM_SHA2_512		0x05
	CRYPTO_ALGOFAM_SHA2_512_224		0x06
	CRYPTO_ALGOFAM_SHA2_512_256		0x07
	CRYPTO_ALGOFAM_SHA3_224		0x08
	CRYPTO_ALGOFAM_SHA3_256		0x09
	CRYPTO_ALGOFAM_SHA3_384		0x0A
	CRYPTO_ALGOFAM_SHA3_512		0x0B
	CRYPTO_ALGOFAM_SHA3_SHAKE128		0x0C
	CRYPTO_ALGOFAM_SHA3_SHAKE256		0x0D
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00177 :**		
Name	CsmRandomGenerateAlgorithmFamilyCustom		
Parent Container	CsmRandomGenerateConfig		
Description	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		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00107 :**	
Name	CsmRandomGenerateAlgorithmMode	
Parent Container	CsmRandomGenerateConfig	
Description	Determines the algorithm mode used for the crypto service	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	CRYPTO_ALGOMODE_CMAC	0x10
	CRYPTO_ALGOMODE_CTRDRBG	0x12
	CRYPTO_ALGOMODE_CUSTOM	0xFF
	CRYPTO_ALGOMODE_GMAC	0x11
	CRYPTO_ALGOMODE_HMAC	0x0f
	CRYPTO_ALGOMODE_NOT_SET	0x00
	CRYPTO_ALGOMODE_SIPHASH_2_4	0x17
	CRYPTO_ALGOMODE_SIPHASH_4_8	0x18

Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00180 :**		
Name	CsmRandomGenerateAlgorithmModeCustom		
Parent Container	CsmRandomGenerateConfig		
Description	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.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00178 :**		
Name	CsmRandomGenerateAlgorithmSecondaryFamily		
Parent Container	CsmRandomGenerateConfig		
Description	Determines the algorithm family used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x00	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00179 :**
Name	CsmRandomGenerateAlgorithmSecondaryFamilyCustom
Parent Container	CsmRandomGenerateConfig
Description	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 Csm RandomAlgorithmSecondaryFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00106 :**		
Name	CsmRandomGenerateResultLength		
Parent Container	CsmRandomGenerateConfig		
Description	Size of the random generate key in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.32 CsmJobKeySetValid

SWS Item	**ECUC_Csm_00196 :**
Container Name	CsmJobKeySetValid
Description	Configurations of KeySetValid primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmJobKeySetValidConfig	1	Container for configuration of a CSM key set valid operation. The container name serves as a symbolic name for the identifier of a key configuration.

10.2.33 CsmJobKeySetValid

SWS Item	**ECUC_Csm_00196 :**
Container Name	CsmJobKeySetValid
Description	Configurations of KeySetValid primitives
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmJobKeySetValidConfig	1	Container for configuration of a CSM key set valid operation. The container name serves as a symbolic name for the identifier of a key configuration.

10.2.34 CsmCallbacks

SWS Item	**ECUC_Csm_00008 :**
Container Name	CsmCallbacks
Description	Container for callback function configurations
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmCallback	0..*	Container for configuration of a callback function

10.2.35 CsmCallback

SWS Item	**ECUC_Csm_00109 :**		
Container Name	CsmCallback		
Description	Container for configuration of a callback function		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Configuration Parameters			

SWS Item	**ECUC_Csm_00110 :**		
Name	CsmCallbackFunc		
Parent Container	CsmCallback		
Description	Callback function to be called if an asynchronous operation has finished. The corresponding job has to be configured to be processed asynchronously.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00111 :**		
Name	CsmCallbackId		
Parent Container	CsmCallback		
Description	Identifier of the callback function. The set of actually configured identifiers shall be consecutive and gapless.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*