Document Title	Specification of Crypto Service Manager
Document Category	Software specification (SWS)
Document UID	402
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Restructure and streamline the configuration model • Remove queue-based job processing model (removal of CsmQueue container and related parameters) • Remove the Csm_GetVersionInfo API and its associated configuration parameter • Editorial changes
2025-11-27	R25-11	AUTOSAR Release Management	• Add Crypto Driver profiles for KeyM Certificate Elements • Extend definition of key elements for other BSW modules • Editorial changes
2024-11-27	R24-11	AUTOSAR Release Management	• Add configuration, cryptographic primitives and schemes for AES KeyWrap and KeyUnwrap `Csm_JobKeyWrap()` `Csm_JobKeyUnwrap()` • Add CRYPTO_E_KEY_EMPTY error for `Csm_RandomSeed()` and CsmJobRandomSeed service interface • Add CRYPTO_E_CUSTOM_ERROR error for reporting custom processing failure • Editorial changes



△

2023-11-23	R23-11	AUTOSAR Release Management	• Add AES Key Wrap support • Remove inconsistency in parameter types and const type modifier • Remove key location check in `Csm_KeyCopy()` • Editorial changes
2022-11-24	R22-11	AUTOSAR Release Management	• New Feature: Addition of the `CsmCustom` service for vendor customized security services. • Removal of certificate functionality (moved to KeyM and replaced by `CsmCustom` services) • Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Harmonize definition of CRYPTO_ALGOMODE between CryptoDrv and Csm • Added key format description in CSM/Crypto Driver for SHE-keys • Added Clarification on seeding and generation of random numbers in the crypto stack • removed superfluous parameter keyId in CsmJobXXX interface operations • Editorial changes
2020-11-30	R20-11	AUTOSAR Release Management	• New feature: Save and Restore Context for running crypto operation • New feature: `KeyGetStatus` and `KeySetInvalid` • Updated Algorithm Families • Support of Multicore • Removing inconsistency in parameter names. • Editorial changes

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Bringing return values of all services and interfaces to one line • added functionality and description of elliptic curves • Callback notification modified • Editorial changes • Changed Document Status from Final to published
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
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

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
2 Acronyms and Abbreviations 10
3 Related documentation 12
3.1 Input documents & related standards and norms . . . 12
3.2 Related specification . . . 12
4 Constraints and assumptions 13
4.1 Limitations . . . 13
4.2 Applicability to car domains . . . 13
4.3 Security Implications . . . 13
5 Dependencies to other modules 14
6 Requirements Tracing 15
7 Functional specification 16
7.1 Basics Architecture Guidelines . . . 16
7.2 Multicore . . . 18
7.3 General Behavior . . . 18
7.3.1 Job-based Operation . . . 19
7.3.1.1 Configuration . . . 20
7.3.2 Design Notes . . . 20
7.3.2.1 General . . . 20
7.3.2.2 CSM module startup . . . 22
7.3.2.3 Crypto Services . . . 23
7.3.2.4 Key Management . . . 23
7.3.2.5 Redirection of Input and/or Output of Crypto Jobs . . . 27
7.3.2.6 Job context interface . . . 28
7.4 Error Classification . . . 29
7.4.1 Development Errors . . . 29
7.4.2 Runtime Errors . . . 29
7.4.3 Production Errors . . . 29
7.5 Error Detection . . . 30
7.6 Security Events . . . 30
8 API specification 31
8.1 Imported types . . . 31
8.2 Type definitions . . . 31
8.2.1 Symbol definitions . . . 32
8.3 Function definitions . . . 33
8.3.1 General Interface . . . 33
8.3.2 Hash Interface . . . 33
8.3.3 MAC Interface . . . 34

8.3.4 CipherInterface . . . 36
8.3.5 Authenticated Encryption with Associated Data (AEAD) Interface . . 38
8.3.6 Signature Interface . . . 40
8.3.7 Random Interface . . . 42
8.3.8 Key Management Interface . . . 43
8.3.8.1 Key Setting Interface . . . 43
8.3.8.2 Key Status Interface . . . 45
8.3.8.3 Key Extraction Interface . . . 46
8.3.8.4 Key Copying Interface . . . 47
8.3.8.5 Key Generation Interface . . . 50
8.3.8.6 Key Derivation Interface . . . 51
8.3.8.7 Key Exchange Interface . . . 52
8.3.9 Cryptographic Primitives and Schemes . . . 55
8.3.10 Context Save and Restore . . . 60
8.3.11 Job Cancellation Interface . . . 62
8.3.12 Custom Interface . . . 63
8.3.12.1 Custom Service Interface . . . 63
8.3.12.2 Custom Sync Interface . . . 64
8.3.13 Callback notifications . . . 65
8.4 Scheduled functions . . . 65
8.5 Expected interfaces . . . 66
8.5.1 Mandatory interfaces . . . 66
8.5.2 Optional interfaces . . . 67
8.5.3 Configurable interfaces . . . 67
8.6 Service Interfaces . . . 68
8.6.1 Client-Server-Interfaces . . . 68
8.6.2 Client-Server Interfaces (DATA_REFERENCES) . . . 86
8.6.3 Client-Server-Interfaces (Key Management) . . . 101
8.6.4 Client-Server-Interfaces (Context Service) . . . 110
8.6.5 Client-Server-Interface Callbacks . . . 112
8.6.6 Implementation Data Types . . . 112
8.6.7 Ports . . . 124

9 Sequence diagrams 127
9.1 Asynchronous Calls . . . 127
9.2 Synchronous Calls . . . 127

10 Configuration specification 129
10.1 How to read this chapter . . . 129
10.2 Containers and configuration parameters . . . 129
10.2.1 Csm . . . 130
10.2.2 CsmGeneral . . . 131
10.2.3 CsmChannels . . . 132
10.2.4 CsmChannel . . . 133

10.2.5 CsmMainFunction . . . 134
10.2.6 CsmJobs . . . 136
10.2.7 CsmJob . . . 137
10.2.8 CsmKeys . . . 141
10.2.9 CsmKey . . . 142
10.2.10 CsmPrimitives . . . 143
10.2.11 CsmPrimitive . . . 144
10.2.12 CsmCallbacks . . . 148
10.2.13 CsmCallback . . . 148
10.2.14 CsmInputRedirectionSets . . . 149
10.2.15 CsmInputRedirectionSet . . . 149
10.2.16 CsmInputRedirection . . . 150
10.2.17 CsmOutputRedirectionSets . . . 152
10.2.18 CsmOutputRedirectionSet . . . 152
10.2.19 CsmOutputRedirection . . . 153
10.3 Published Information . . . 155

A Change history of AUTOSAR traceable items 156
A.1 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 156
A.1.1 Added Specification Items in R25-11 . . . 156
A.1.2 Changed Specification Items in R25-11 . . . 156
A.1.3 Deleted Specification Items in R25-11 . . . 156
A.2 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 156
A.2.1 Added Specification Items in R26-11 . . . 156
A.2.2 Changed Specification Items in R26-11 . . . 157
A.2.3 Deleted Specification Items in R26-11 . . . 157

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the software module Crypto Service Manager (CSM).

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

The glossary below includes acronyms and abbreviations relevant to the Crypto Service Manager module that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	*Description:*
AEAD	Authenticated Encryption with Associated Data
CDD	Complex Device Driver
CSM	Crypto Service Manager
CRYIF	Crypto Interface [2]
CRYPTO	Crypto Driver [3]
DET	Default Error Tracer [4]
HSM	Hardware Security Module
HW	Hardware
SHE	Security Hardware Extension
SW	Software

Table 2.1: Acronyms and abbreviations used in the scope of this Document

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. Key elements from different keys have different memory area (both NV and RAM area).	
Job	A Job is the statically configured combination of e.g. a primitive reference, a channel, a priority, an optional key, etc., accessible by the user via its id.	
Channel	A channel is the statically configured path of a Job provided by the Crypto Service Manager via the Crypto Interface to a specific Crypto Driver Object.	
Primitive	A primitive is an instance of a configured cryptographic algorithm realized in a Crypto Driver Object. Among others it refers to a functionality provided by the CSM to the application, the concrete underlining 'algorithmfamily' (e.g. AES, MD5, RSA, etc.), and a 'algorithmmode' (e.g. ECB, CBC, etc).	
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.

△

Terms:	*Description:*	
	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.
Service	A service shall be understand as defined in the FO_TR_Glossary document: A service is a type of operation that has a published specification of interface and behavior, involving a contract between the provider of the capability and the potential clients.	

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] Specification of Crypto Interface
AUTOSAR_CP_SWS_CryptoInterface

[3] Specification of Crypto Driver
AUTOSAR_CP_SWS_CryptoDriver

[4] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[5] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[6] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[7] IETF RFC 5639 – Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation, 2010

[8] IETF RFC 6637 – Elliptic Curve Cryptography (ECC) in OpenPGP, 2012

[9] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [5, 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.⌋

[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.⌋

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.

6 Requirements Tracing

The following tables reference the requirements specified in [6] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

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]	Callback Function Return Types for AUTOSAR BSW	[SWS_Csm_00970] [SWS_Csm_00971]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[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_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Csm_00186] [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_BSW_00438]	Configuration data shall be defined in a structure	[SWS_Csm_01085]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Basics Architecture Guidelines

The architecture of the AUTOSAR Basic Software consists of several layers which can be seen in Figure 7.1.

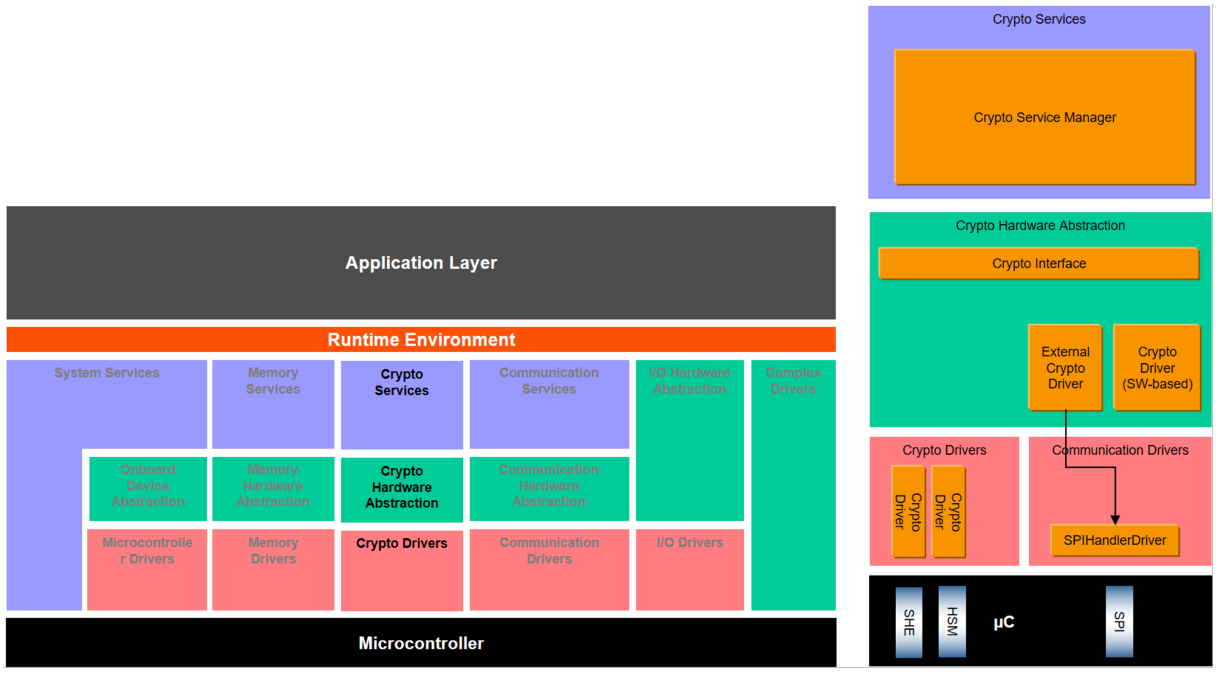


Figure 7.1: AUTOSAR Layered View

The Service Layer (purple in Figure 7.1) is the highest layer within the Basic Software. Its task is to provide basic services for applications and basic software modules, i.e. it offers the most relevant functionalities for application software and basic software modules.

The ECU Abstraction Layer (green in Figure 7.1) interfaces the drivers of the Microcontroller Abstraction Layer including an abstracted access of multiple instances of same drivers towards the Service Layer.

The Microcontroller Abstraction Layer (red in Figure 7.1) is the lowest software layer of the Basic Software. It contains internal drivers, which are typically software modules with direct access to the Microcontroller and internal peripherals.

Within this AUTOSAR Layered Basic Software Architecture, the CSM module, as specified in this document, is located in the Service Layer. Here it provides basic services in the context of cryptography to application software and other basic software modules. The services are

- job-based cryptographic primitives and job-based management of key material
- not job-based management of key material

They are based on static configurations of keys and/or keys, primitives, channels, (optional) input-/output-redirections, callbacks (for asynchronous jobs) and their combination to jobs, extended by a jobs id, priority, processing mode (synchronous or asynchronous), etc. Requests of specific service operations by applications or other basic software modules are directly forwarded by the CSM to the CRYIF.

The CRYIF in turn, which is located on the ECU Abstraction Layer, forwards requests of service operations to

- the CRYPTO driver and its Crypto Driver Object that is statically assigned to the channel, the job mentioned by the request belongs to
- the CRYPTO driver that is statically assigned to the key mentioned by the request.

The multi-instance capable CRYPTO driver is located on the Microcontroller Abstraction Layer and can rely on software libraries or on hardware cryptography modules (e.g. SHE, HSM, TPM). Hence the excecution of any cryptographic primitive or key management functionality is always performed in/by the CRYPTO driver, not by the CSM or CRYIF. The keys, composed of key elements, used in the execution of cryptographic primitives and whose material can be created, generated or modified by job-based or not job-based key management operations are always located in the CRYPTO driver itself. CSM, CRYIF, applications and other basic softwar modules only use direct or indirect references to these keys in the CRYPTO.

Beside the actual realization of cryptographic primitives and key management operations, the CRYPTO driver also realizes Crypto Driver Objects; representations of independent cryptographic execution units (software libraries or on hardware cryptography module based) for job-based requests of cryptographic primitives and key management operations. Each Crypto Driver Object contains a single priority based queue for asynchronous job-based requests and is the endpoint of a channel through the whole Crypto Stack starting in the CSM. Figure 7.2 provides an example of this.

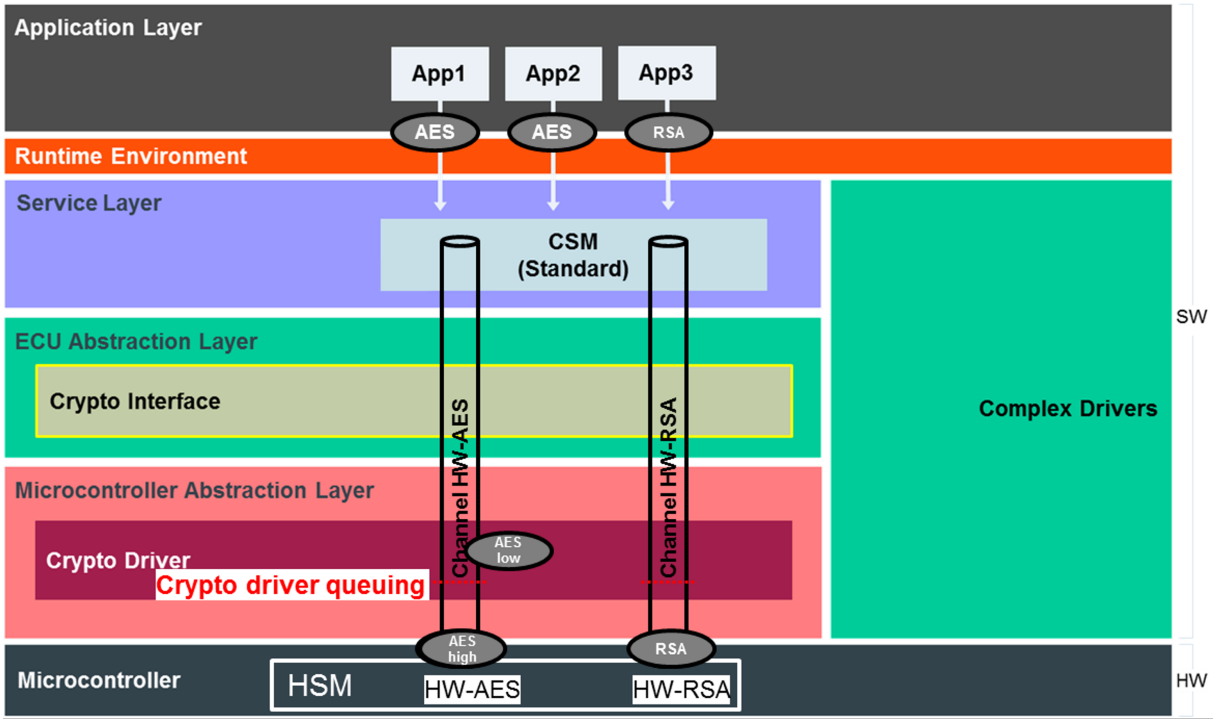


Figure 7.2: AUTOSAR Layered View with channels

In Figure 7.2 a HSM is illustrated with HW-accelerators for AES and RSA, as well as a CRYPTO driver with two Crypto Driver Objects, each corresponding to one of the HW-accelerators and each the endpoint of a channel. Furthermore the Crypto Driver Objects are processing a crypto job (AES-high and RSA) each, while the queue of the leftmost Crypto Driver Object contains another job (AES-low). If the HW-AES of the HSM finished the AES-high job, AES-low job will be processed as next one.

7.2 Multicore

[SWS_Csm_91066] ⌈The Csm module shall apply appropriate mechanisms to allow calls of `Csm_<Service>` API from partitions its `CsmJob`s are assigned to.⌋

[SWS_Csm_91076] Job partition assignment through the channel main function ⌈The `CsmMainFunction` referenced to the `CsmChannel` assigned to a `CsmJob` shall define the partition, where the `CsmJob` is assigned to.⌋

7.3 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 maintained by CSM.⌋

[SWS_Csm_00017] ⌈If the processing of a job-based cryptographic primitive or the job-based management of key material is requested from the CSM module, then the job request shall call `CryIf_ProcessJob` and pass on the return value.⌋

[SWS_Csm_91077] Forwarding of non-job-based key management requests to CRYIF ⌈If the not job-based management of key material is requested from the CSM module, then the request shall call the corresponding CRYIF API and pass on the return value.⌋

7.3.1 Job-based 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.

⌋

[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 `CRYPTO_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.

[SWS_Csm_91078] Forwarding of asynchronous job requests to the Crypto Stack ⌈If the processing of a CsmJob with any valid operation mode and configured asynchronous processing type is requested from the CSM, the interface functions shall only hand over the necessary information to the underlying Crypto Stack modules.⌋

[SWS_Csm_91079] Callback notification after asynchronous job processing ⌈If the processing of a CsmJob with any valid operation mode and configured asynchronous processing type has been processed, the requesting user shall be notified by calling the configured callback function from the CsmJobs configuration.⌋

7.3.1.1 Configuration

[SWS_Csm_91080] Constant job configuration instance ⌈All configuration dependent information of a job shall be realized as a constant instance of type `Crypto_JobConfigType`.⌋

[SWS_Csm_91081] Constant input redirection configuration array ⌈All configuration dependent information related to the inputs of a redirection (CsmInOutRedirection) referenced by at least one job shall be realized as a constant array of size CRYPTO_MAX_INPUTS and type `Crypto_DataRedirectionType`.⌋

[SWS_Csm_91082] Constant output redirection configuration array ⌈All configuration dependent information related to the outputs of a redirection (CsmInOutRedirection) referenced by at least one job shall be realized as a constant array of size CRYPTO_MAX_OUTPUTS and type `Crypto_DataRedirectionType`.⌋

[SWS_Csm_91083] Variable job run-time data instance ⌈All run-time data of a job shall be realized as a variable instance of type `Crypto_JobActionType`.⌋

[SWS_Csm_91084] Service-dependent job input data array ⌈All data related to input data to a job shall be realized as an array of type `Crypto_DataInputType` and a size depending on the job's service, that is referenced into the job's instance of type `Crypto_JobActionType`.⌋

[SWS_Csm_91085] Service-dependent job output data array ⌈All data related to output data of a job shall be realized as an array of type `Crypto_DataOutputType` and a size depending on the job's service, that is referenced into the job's instance of `Crypto_JobActionType`.⌋

[SWS_Csm_00032] ⌈If the asynchronous processing type is chosen for a job the job configuration shall contain a callback function.⌋

7.3.2 Design Notes

7.3.2.1 General

The CSM provides two services:

1. the crypto services itself and
2. key management.

The Crypto services are usable via Csm jobs and service related APIs. The key management is usable via Csm keys and related APIs.

It is characteristic for Csm jobs requested via a service API that they expect inputs and generate outputs.

[SWS_Csm_91086] Service-dependent allocation of job inputs and outputs ⌈The number of Inputs and Outputs of a specific job memory shall be allocated for, shall depend on the job's service itself.⌋

The number of Inputs and Outputs per service is specified in SWS_Crypto_00071 and SWS_Crypto_00073.

[SWS_Csm_91087] Separate arrays for job inputs and outputs ⌈The Inputs and Outputs of a job shall be configured in two individual, referenced arrays, one for the inputs and another one for the outputs.⌋

Note: It is also possible to use keys as inputs or outputs to a Csm job.

[SWS_Csm_91088] Separate arrays for job input and output redirections ⌈The redirections of Inputs and Outputs of a job shall be configured in two individual, referenced arrays, one for the inputs and another one for the outputs.⌋

[SWS_Csm_91089] Reuse of input and output redirection configurations between jobs ⌈The redirection configurations for Inputs and Outputs for jobs shall be reusable between different jobs.⌋

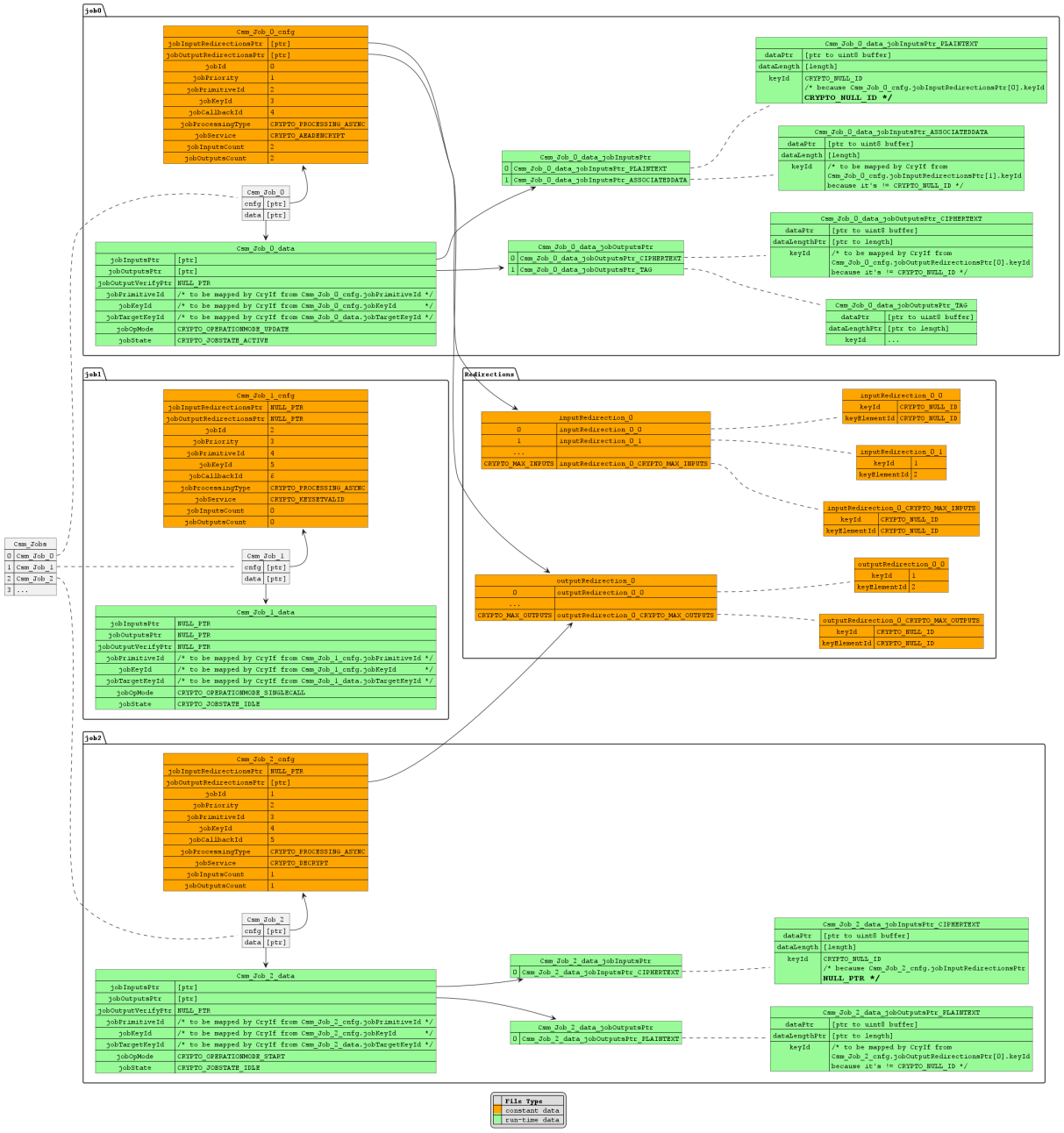


Figure 7.3: Structure of Input- and Output-Redirection

7.3.2.2 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.

7.3.2.3 Crypto Services

7.3.2.3.1 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_01055] ⌈Only the service operations HASH, MACGENERATE, MACVERIFY, ENCRYPT, DECRYPT, AEAD_ENCRYPT, AEAD_DECRYPT, SIGNATUREGENERATE, SIGNATUREVERIFY shall support the operation mode `START`, `UPDATE` and `FINISH` as specified from the API. For all other service operations, the CSM shall set the operation mode to `CRYPTO_OPERATIONMODE_SINGLECALL`, even if the API does not provide an operation mode.⌋

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 pointer to a `Crypto_JobActionType` sub structure, 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_JobActionType`, can be found in [SWS_Crypto_00073] of the Crypto Driver specification.

7.3.2.4 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 Crypto_GeneralTypes.h:

[SWS_Csm_01022] ⌈

Crypto Service:	Key element:	Key element Name:	Key element ID:
MAC	Key Material	CRYPTO_KE_MAC_KEY	1
	Proof (SHE)	CRYPTO_KE_MAC_PROOF	2
	Seed	CRYPTO_KE_KEYGENERATE_SEED	16
Signature	Key Material	CRYPTO_KE_SIGNATURE_KEY	1
	ECC curve type	CRYPTO_KE_SIGNATURE_CURVETYPE	29
Random	Seed State	CRYPTO_KE_RANDOM_SEED_STATE	3
	Algorithm	CRYPTO_KE_RANDOM_ALGORITHM	4
Cipher/AEAD	Key Material	CRYPTO_KE_CIPHER_KEY	1
	Proof (SHE)	CRYPTO_KE_CIPHER_PROOF	2
	Init Vector	CRYPTO_KE_CIPHER_IV	5
	2nd Key Material	CRYPTO_KE_CIPHER_2NDKEY	7
Key Exchange	Base	CRYPTO_KE_KEYEXCHANGE_BASE	8
	Private Key	CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9
	Own Public Key	CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY	10
	Shared Value	CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE	1
	Algorithm	CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12
	ECC curve type	CRYPTO_KE_KEYEXCHANGE_CURVETYPE	29
Key Derivation	Password	CRYPTO_KE_KEYDERIVATION_PASSWORD	1
	Salt	CRYPTO_KE_KEYDERIVATION_SALT	13
	Iterations	CRYPTO_KE_KEYDERIVATION_ITERATIONS	14
	Algorithm	CRYPTO_KE_KEYDERIVATION_ALGORITHM	15
	ECC curve type	CRYPTO_KE_KEYDERIVATION_CURVETYPE	29
Key Generate	Key Material	CRYPTO_KE_KEYGENERATE_KEY	1
	Seed	CRYPTO_KE_KEYGENERATE_SEED	16
	Algorithm	CRYPTO_KE_KEYGENERATE_ALGORITHM	17
	ECC curve type	CRYPTO_KE_KEYGENERATE_CURVETYPE	29
Key Wrap / Key Unwrap	Key to be wrapped / Unwrapped key	CRYPTO_KE_KEYWRAP_KEY	1
	Header Information	CRYPTO_KE_KEYWRAP_HEADER	2

The above standardized key elements used for Csm services and key management shall be made available via Crypto_GeneralTypes.h.

⌋

The key elements indices of [SWS_Csm_01022] 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.⌋

[SWS_Csm_00953] ⌈The following key formats are available:

See [SWS_Csm_01096].

A binary Octet is the integer representation in base 256. A large value can be splitted into his factors:

$$x = x_{xLen-1} * 256^{xLen-1} + x_{xLen-2} * 256^{xLen-2} + \ldots + x_1 * 256 + x_0 \; where \; 0 \leq x_i < 256$$

Let the Octet Xi have the integer value x_{xLen-i} for $1 \leq i \leq xLen$. 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.⌋

[SWS_Csm_01096] Available key formats ⌈

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.

⌋

[SWS_Csm_00952] ⌈Vendor specific keyElementIds should start 1000 to avoid interferences with future extended versions of the Crypto Stack. KeyElementIds specific to particular BSW modules (e.g. MACSec) should start at 9000 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.

[SWS_Csm_01092] ⌈If a cryptographic primitive uses elliptic curve algorithm but the concrete curve parameter cannot sufficiently specified by its algorithm families and its algorithm mode, an additional key element of type `CRYPTO_KE_xxxxx_CURVETYPE` shall be used to provide the required information. This information is set at runtime through the key element interface. The data of the key element shall be set with its object identifier follows the format defined in [7] and [8].⌋

```
1    const uint8 * EccKey = { 0x12, 0x23, 0x34, ... };
2    // The required key value.
3    // According to RFC5639:
```

```
4    // {iso(1) identified-organization(3) teletrust(36) algorithm(3)
     signatureAlgorithm(3) ecSign(2) ecStdCurvesAndGeneration(8)
     ellipticCurve(1)}
5    brainpoolP160r1(1)
6
7    const uint8 * EccType = { 1, 3, 36, 3, 3, 2, 8, 1, 1 };
8    //OID definition of ECC Brainpool 160 P1
9
10   Csm_KeyElementSet(MyEccKeyId, CRYPTO_KE_SIGNATURE_KEY, EccKey, sizeof(
     EccKey));
11   Csm_KeyElementSet(MyEccKeyId, CRYPTO_KE_SIGNATURE_CURVETYPE, EccType,
     sizeof(EccType));
12   Csm_KeySetValid(MyEccKeyId);
```

Listing 7.1: Definition for an ECC Brainpool 160 P1 key used for signature generation.

7.3.2.4.1 Crypto Driver profiles

The Crypto Driver can support vendor specific custom services and custom synchronous API functions, triggered by `Csm_CustomService` and `Csm_CustomSync` (Crypto_CustomSync). In order to align the realization also for other BSW modules for a particular use case, a definition of key elements is required. This is defined by the definition of profiles for particular use case. The following defines groups of key element ids as they are used in a particular Crypto Driver profile. They shall be made available via Crypto_GeneralTypes.h

[SWS_Csm_00960] Mapping table for KeyM Certificate Elements ⌈Profile: KeyM and certificate management⌋

KeyMCertificateElementOf-Structure	key element Name	key element ID	Mandatory
Certificate	CRYPTO_KE_CERTIFICATE_DATA	0	x
Format	CRYPTO_KE_CERTIFICATE_PARSING_FORMAT	18	
CertificateVersionNumber	CRYPTO_KE_CERTIFICATE_VERSION	20	
Serial Number	CRYPTO_KE_CERTIFICATE_SERIALNUMBER	21	
CertificateSignatureAlgo-rithm	CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM	22	
CertificateIssuerName	CRYPTO_KE_CERTIFICATE_ISSUER	23	
CertificateValidityPeriodNot-Before	CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE	24	
CertificateValidityPeriodNo-tAfter	CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER	25	
CertificateSubjectName	CRYPTO_KE_CERTIFICATE_SUBJECT	26	
CertificateSubjectPublicKey-Info_SubjectPublicKey	CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY	1	
Extensions	CRYPTO_KE_CERTIFICATE_EXTENSIONS	27	
Signature	CRYPTO_KE_CERTIFICATE_SIGNATURE	28	
CertificateIssuerUniqueI-dentifier	CRYPTO_KE_CERTIFICATE_ISSUER_UNIQUE_ IDENTIFIER	29	

▽

△

CertificateSignatureAlgorithmID	CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM_ID	30	
CertificateSubjectPublicKeyInfo_PublicKeyAlgorithm	CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLICKEYINFO_PUBLICKEYALGORITHM	31	
CertificateSubjectUniqueIdentifier	CRYPTO_KE_CERTIFICATE_SUBJECT_UNIQUE_IDENTIFIER	32	
RevokedCertificates	CRYPTO_KE_CERTIFICATE_REVOKED_CERTIFICATES	33	
CertificateSubjectAuthorization	CRYPTO_KE_CERTIFICATE_SUBJECT_AUTHORIZATION	34	

Certificate element names and Ids

7.3.2.5 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 (`CsmJobInputRedirectionSetRef` [ECUC_Csm_00397] or `CsmJobOutputRedirectionSetRef` [ECUC_Csm_00398] 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_91090] Null redirection pointers for jobs without redirection ⌈If input and output redirection is not used at all for a specific job (neither `CsmJobInputRedirectionSetRef` [ECUC_Csm_00397] nor `CsmJobOutputRedirectionSetRef` [ECUC_Csm_00398] is existing), then `Crypto_JobConfigType->jobInputRedirectionsPtr` and `Crypto_JobConfigType->jobOutputRedirectionsPtr` shall both be set to NULL_PTR.⌋

[SWS_Csm_91091] Redirection pointers for jobs with input redirection only ⌈If input redirection is used for a specific job but no output redirection, then `Crypto_JobConfigType->jobInputRedirectionsPtr` shall reference an array of size CRYPTO_MAX_INPUTS and type `Crypto_DataRedirectionType` and `Crypto_JobConfigType->jobOutputRedirectionsPtr` shall be set to NULL_PTR.⌋

[SWS_Csm_91092] Redirection pointers for jobs with output redirection only ⌈If output redirection is used for a specific job but no input redirection, then `Crypto_JobConfigType->jobOutputRedirectionsPtr` shall reference an array of size CRYPTO_MAX_OUTPUTS and type `Crypto_DataRedirectionType` and `Crypto_JobConfigType->jobInputRedirectionsPtr` shall be set to NULL_PTR.⌋

[SWS_Csm_91093] Configured key identifiers for redirected inputs ⌈If input redirection is used for specific inputs of a specific job (`CsmJobInputRedirectionSetRef` [ECUC_Csm_00397] is existing and at least one input redirection is configured),

then the entries in the input redirection array corresponding to redirected inputs shall specify the configured keyIds and keyElementIds.⌋

[SWS_Csm_91094] Null key identifiers for non-redirected inputs ⌈If input redirection is used for specific inputs of a specific job (CsmJobInputRedirectionSetRef [ECUC_Csm_00397] is existing and at least one input redirection is configured), then the entries in the input redirection array corresponding to NOT redirected inputs shall specify their keyIds and keyElementIds to CRYPTO_NO_ID.⌋

[SWS_Csm_91095] Configured key identifiers for redirected outputs ⌈If output redirection is used for specific outputs of a specific job (CsmJobOutputRedirectionSetRef [ECUC_Csm_00398] is existing and at least one output redirection is configured), then the entries in the output redirection array corresponding to redirected outputs shall specify the configured keyIds and keyElementIds.⌋

[SWS_Csm_91096] Null key identifiers for non-redirected outputs ⌈If output redirection is used for specific outputs of a specific job (CsmJobOutputRedirectionSetRef [ECUC_Csm_00398] is existing and at least one output redirection is configured), then the entries in the output redirection array corresponding to NOT redirected outputs shall specify their keyIds and keyElementIds to CRYPTO_NO_ID.⌋

7.3.2.6 Job context interface

The job context interface allows to save or restore the context data of the workspace for a specific crypto service from the crypto driver. This allows to store all dynamically created data within a crypto driver so that it can later be restored to continue this operation at the exact point where the context snapshot was taken.

Key element data are not affected. This means, that key elements are not part of the context data and must be set or read by the key element interface separately if necessary.

[SWS_Csm_91069] ⌈In addition to the standard error detection, on a call to Csm_SaveContextJob or Csm_RestoreContextJob the CSM shall check if the related job is currently active. If so, the operation shall continue as specified. Otherwise, the function shall immediately return with E_NOT_OK.⌋

[SWS_Csm_91070] ⌈On a call to Csm_SaveContextJob or Csm_RestoreContextJob the CSM shall check if the related job is currently actively processing data, means the CSM has scheduled an operation related to this job to the driver and is waiting for a response. In this case, the function shall immediately return with E_NOT_OK.⌋

[SWS_Csm_91071] ⌈On a call to Csm_RestoreContextJob or Csm_SaveContextJob the CSM shall check if the job references to either of the service CRYPTO_HASH, CRYPTO_MACGENERATE, CRYPTO_MACVERIFY, CRYPTO_

ENCRYPT, CRYPTO_DECRYPT, CRYPTO_AEADENCRYPT, CRYPTO_AEADDECRYPT, CRYPTO_SIGNATUREGENERATE or CRYPTO_SIGNATUREVERIFY. If so, the operation shall continue as specified. Otherwise, the operation shall not be performed and CSM_E_SERVICE_TYPE shall be reported to the DET when CsmDevErrorDetect is true.⌋

7.4 Error Classification

Chapter [5, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.4.1 Development Errors

[SWS_Csm_91004] Definition of development errors in module Csm ⌈

Type of error	*Related error code*	*Error value*
API request called with invalid parameter (Nullpointer)	CSM_E_PARAM_POINTER	0x01
Csm Configuration ID 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
Mismatch between the called API request and the service type of the job	CSM_E_SERVICE_TYPE	0x09

⌋

7.4.2 Runtime Errors

There are no runtime errors.

7.4.3 Production Errors

There are no production errors.

7.5 Error Detection

[SWS_Csm_91008] ⌈While the CSM is not initialized and any function of the CSM API is called, except of `Csm_Init`, 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 ID handle in its interface is called and the ID handle 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_01091] ⌈If a CSM API with a job handle (called jobId) in its interface is called and the `Crypto_ServiceInfoType` of the job does not match the requested service, the operation shall not be performed and `CSM_E_SERVICE_TYPE` shall be reported to the DET when `CsmDevErrorDetect` is true.⌋

7.6 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[SWS_Csm_00068] Definition of imported datatypes of module Csm ⌈

Module	*Header File*	*Imported Type*
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

[SWS_Csm_91043] Definition of Std_ReturnType-extension for module Csm ⌈

Range	CRYPTO_E_BUSY	0x02	The service request failed because the service is still busy
	CRYPTO_E_ENTROPY_EXHAUSTED	0x04	The service request failed because the entropy of the random number generator is exhausted
	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 at least one required key element 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_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.
	CRYPTO_E_CUSTOM_ERROR	0x0E	Custom processing failed.
Description	–		
Available via	Crypto_GeneralTypes.h		

⌋

[SWS_Csm_01085] Definition of datatype Csm_ConfigType

Upstream requirements: SRS_BSW_00438

⌈

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

⌋

8.2.1 Symbol definitions

[SWS_Csm_91116] Definition of symbol CRYPTO_MAX_INPUTS ⌈

Name	CRYPTO_MAX_INPUTS
Kind	Symbol
Base Type	uint8
Value	0x03U
Description	Maximum number of inputs to a service supported by the Crypto stack.
Available via	Crypto_GeneralTypes.h

⌋

[SWS_Csm_91117] Definition of symbol CRYPTO_MAX_OUTPUTS ⌈

Name	CRYPTO_MAX_OUTPUTS
Kind	Symbol
Base Type	uint8
Value	0x02U
Description	Maximum number of outputs to a service supported by the Crypto stack.
Available via	Crypto_GeneralTypes.h

⌋

[SWS_Csm_91118] Definition of symbol CRYPTO_NO_ID ⌈

Name	CRYPTO_NO_ID
Kind	Symbol
Base Type	uint32
Value	0xFFFFFFFFU
Description	Identifier for the identification of deliberately unconfigured keys, callbacks, etc. of a Csm Job.
Available via	Crypto_GeneralTypes.h

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

[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

[SWS_Csm_00646] Definition of API function Csm_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414

⌈

Service Name	Csm_Init	
Syntax	`void Csm_Init ( const Csm_ConfigType* configPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	configPtr	Pointer to a selected configuration structure
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initializes the CSM module. In configurations, in which Csm is assigned to more than one partition (i.e. Csm_MainFunctions are mapped to partitions), Csm may provide one init function per partition.	
Available via	Csm.h	

⌋

[SWS_Csm_00186]

Upstream requirements: SRS_BSW_00414

⌈The Configuration pointer `configPtr` shall always have a `null` pointer value.⌋

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.⌋

8.3.2 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.

[SWS_Csm_00980] Definition of API function Csm_Hash ⌈

Service Name	Csm_Hash	

Syntax	`Std_ReturnType Csm_Hash (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr`
	`)`	

Service ID [hex]	0x5d	
Sync/Async	Depends on 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. If the provided length information is smaller than the total length of the hash result, the resultPtr will contain the truncated hash result.
Parameters (out)	resultPtr	Contains the pointer to the data where the hash value shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed
Description	Uses the given data to perform the hash calculation and stores the hash.	
Available via	Csm.h	

⌋

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.

[SWS_Csm_00982] Definition of API function Csm_MacGenerate ⌈

Service Name	Csm_MacGenerate	
Syntax	`Std_ReturnType Csm_MacGenerate ( uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* macPtr, uint32* macLengthPtr )`	
Service ID [hex]	0x60	
Sync/Async	Depends on 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. If the provided length information is smaller than the total length of the MAC result, the macPtr will contain the truncated MAC result.
Parameters (out)	macPtr	Contains the pointer to the data where the MAC shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, a key element has the wrong size `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	
Available via	Csm.h	

⌋

[SWS_Csm_01050] Definition of API function Csm_MacVerify ⌈

Service Name	Csm_MacVerify	
Syntax	`Std_ReturnType Csm_MacVerify ( uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, const uint8* macPtr, uint32 macLength, Crypto_VerifyResultType* verifyPtr )`	
Service ID [hex]	0x61	
Sync/Async	Depends on 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	Holds a pointer to the data for which the MAC shall be verified.
	dataLength	Contains the number of data bytes for which the MAC shall be verified.
	macPtr	Holds a pointer to the MAC to be verified.
	macLength	Contains the MAC length in BITS to be verified.
Parameters (inout)	None	
Parameters (out)	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Verifies the given MAC by comparing if the MAC is generated with the given data.	
Available via	Csm.h	

⌋

8.3.4 CipherInterface

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.

[SWS_Csm_00984] Definition of API function Csm_Encrypt ⌈

Service Name	Csm_Encrypt	

Syntax	`Std_ReturnType Csm_Encrypt (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr`
	`)`	

Service ID [hex]	0x5e	
Sync/Async	Depends on 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 result Ptr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Contains the pointer to the data where the encrypted data shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, a key element has the wrong size `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.	
Available via	Csm.h	

⌋

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

[SWS_Csm_00989] Definition of API function Csm_Decrypt ⌈

Service Name	Csm_Decrypt	

Syntax	`Std_ReturnType Csm_Decrypt (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr`
	`)`	

Service ID [hex]	0x5f	
Sync/Async	Depends on 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 result Ptr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	resultPtr	Contains the pointer to the memory location where the decrypted data shall be stored.

▽

△

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, a key element has the wrong size CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Decrypts the given encrypted data and store the decrypted plaintext in the memory location pointed by the result pointer.	
Available via	Csm.h	

⌋

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).

[SWS_Csm_01023] Definition of API function Csm_AEADEncrypt ⌈

Service Name	Csm_AEADEncrypt	

Syntax	`Std_ReturnType Csm_AEADEncrypt (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* plaintextPtr,`
	`  uint32 plaintextLength,`
	`  const uint8* associatedDataPtr,`
	`  uint32 associatedDataLength,`
	`  uint8* ciphertextPtr,`
	`  uint32* ciphertextLengthPtr,`
	`  uint8* tagPtr,`
	`  uint32* tagLengthPtr`
	`)`	

Service ID [hex]	0x62	
Sync/Async	Depends on 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 (inout)	ciphertextLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the ciphertext is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.

▽

△

	tagLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the Tag is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	ciphertextPtr	Contains the pointer to the data where the encrypted data shall be stored.
	tagPtr	Contains the pointer to the data where the Tag shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, a key element has the wrong size `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Uses the given input data to perform a AEAD encryption and stores the ciphertext and the MAC in the memory locations pointed by the ciphertext pointer and Tag pointer.	
Available via	Csm.h	

⌋

[SWS_Csm_01026] Definition of API function Csm_AEADDecrypt ⌈

Service Name	Csm_AEADDecrypt	
Syntax	`Std_ReturnType Csm_AEADDecrypt (uint32 jobId, Crypto_OperationModeType mode, const uint8* ciphertextPtr, uint32 ciphertextLength, const uint8* associatedDataPtr, uint32 associatedDataLength, const uint8* tagPtr, uint32 tagLength, uint8* plaintextPtr, uint32* plaintextLengthPtr, Crypto_VerifyResultType* verifyPtr)`	
Service ID [hex]	0x63	
Sync/Async	Depends on 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 plaintext Ptr. When the request has finished, the actual length of the returned value shall be stored.

▽

△

Parameters (out)	plaintextPtr	Contains the pointer to the data where the decrypted data shall be stored.
	verifyPtr	Contains the pointer to the result of the verification.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, a key element has the wrong size `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Uses the given ciphertext and associated data as input to perform an AEAD decryption and stores the output plaintext in the memory location pointed by the plaintext pointer and stores the verification result in the memory location pointed by the verifyPtr.	
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.

[SWS_Csm_00992] Definition of API function Csm_SignatureGenerate ⌈

Service Name	Csm_SignatureGenerate	

Syntax	`Std_ReturnType Csm_SignatureGenerate (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  uint8* signaturePtr,`
	`  uint32* signatureLengthPtr`
	`)`	

Service ID [hex]	0x76	
Sync/Async	Depends on 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)	signatureLengthPtr	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 result Ptr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	signaturePtr	Contains the pointer to the data where the signature shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, a key element has the wrong size `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.	
Available via	Csm.h	

⌋

[SWS_Csm_00996] Definition of API function Csm_SignatureVerify ⌈

Service Name	Csm_SignatureVerify	

Syntax	`Std_ReturnType Csm_SignatureVerify (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  const uint8* signaturePtr,`
	`  uint32 signatureLength,`
	`  Crypto_VerifyResultType* verifyPtr`
	`)`	

Service ID [hex]	0x64	
Sync/Async	Depends on 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 verified.
	dataLength	Contains the number of data bytes.
	signaturePtr	Holds a pointer to the signature to be verified.
	signatureLength	Contains the signature length in bytes.
Parameters (inout)	None	
Parameters (out)	verifyPtr	Holds a pointer to the memory location, which will hold the result of the signature verification.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, a key element has the wrong size `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Verifies the given signature by checking if it was generated with the given data.	

▽

△

Available via	Csm.h

⌋

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.

Note: How the random generators are seeded is project specific and out of scope of this specification. If applicable, proper seeding actions shall be done prior to request any random numbers.

An ECU-centralized generation of entropy is recommended, to seed random number generators. Especially if no true random number generator (TRNG) in hardware is available for generation of random numbers.

[SWS_Csm_01543] Definition of API function Csm_RandomGenerate ⌈

Service Name	Csm_RandomGenerate	
Syntax	`Std_ReturnType Csm_RandomGenerate ( uint32 jobId, uint8* resultPtr, uint32* resultLengthPtr )`	
Service ID [hex]	0x72	
Sync/Async	Depends on 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 size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned random number shall be stored. If the provided length information is smaller than the total length of the random number result, the resultPtr will contain the truncated random number.
Parameters (out)	resultPtr	Holds a pointer to the memory location which will hold the result of the random number generation.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_ENTROPY_EXHAUSTED`: Request failed, entropy of random number generator is exhausted
Description	Generate a random number and stores it in the memory location pointed by the result pointer.	
Available via	Csm.h	

⌋

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

[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)

For services that have a CRYPTO_KE_*_ALGORITHM key element (see [SWS_Csm_01022]), that key element specifies the algorithm. The key element value is vendor-specific.

8.3.8.1 Key Setting Interface

[SWS_Csm_00957] Definition of API function Csm_KeyElementSet ⌈

Service Name	Csm_KeyElementSet	
Syntax	`Std_ReturnType Csm_KeyElementSet ( uint32 keyId, uint32 keyElementId, const uint8* keyElementPtr, uint32 keyElementLength )`	
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.
	keyElementPtr	Holds the pointer to the key element bytes to be processed.
	keyElementLength	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	

⌋

Note: In case of error observed during Csm_KeyElementSet API call, the status of the key element needs to be considered as unknown. The application could explicitly check the key content status by calling Csm_KeyGetStatus.

[SWS_Csm_01002] ⌈If no errors are detected by Csm, the service Csm_KeyElementSet shall call CryIf_KeyElementSet.⌋

[SWS_Csm_00958] Definition of API function Csm_KeySetValid ⌈

Service Name	Csm_KeySetValid	
Syntax	`Std_ReturnType Csm_KeySetValid ( uint32 keyId )`	
Service ID [hex]	0x67	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	keyId	Holds the identifier of the key for which a new material shall be validated.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, 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.⌋

[SWS_Csm_91075] Definition of API function Csm_KeySetInvalid ⌈

Service Name	Csm_KeySetInvalid
Syntax	`Std_ReturnType Csm_KeySetInvalid ( uint32 keyId )`
Service ID [hex]	0x85

▽

△

Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	keyId	Holds the identifier of the key which shall be invalidated.
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 status to invalid. The key cannot be used any longer for cryptographic operations until it has been set to valid state again.	
Available via	Csm.h	

⌋

[SWS_Csm_91050] ⌈If no errors are detected by Csm, the service `Csm_KeySetInvalid` shall call `CryIf_KeySetInvalid`.⌋

8.3.8.2 Key Status Interface

[SWS_Csm_91047] Definition of API function Csm_KeyGetStatus ⌈

Service Name	Csm_KeyGetStatus	

Syntax	`Std_ReturnType Csm_KeyGetStatus (`
	`  uint32 keyId,`
	`  Crypto_KeyStatusType* keyStatusPtr`
	`)`	

Service ID [hex]	0x83	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	keyId	Holds the identifier of the key for which the key state shall be returned.
Parameters (inout)	None	
Parameters (out)	keyStatusPtr	Contains the pointer to the data where the status of the key shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed
Description	Returns the key state of the key identified by keyId.	
Available via	Csm.h	

⌋

[SWS_Csm_91048] ⌈If no errors are detected by Csm, the service `Csm_KeyGetStatus` shall call `CryIf_KeyGetStatus`.⌋

8.3.8.3 Key Extraction Interface

[SWS_Csm_00959] Definition of API function Csm_KeyElementGet ⌈

Service Name	Csm_KeyElementGet	
Syntax	`Std_ReturnType Csm_KeyElementGet ( uint32 keyId, uint32 keyElementId, uint8* keyElementPtr, uint32* keyElementLengthPtr )`	
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)	keyElementLengthPtr	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 keyElementPtr. When the request has finished, the actual size of the written input bytes shall be stored.
Parameters (out)	keyElementPtr	Holds the pointer to the memory location where the key element shall be copied to.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, Crypto Driver Object is busy `CRYPTO_E_KEY_NOT_AVAILABLE`: Request failed, the requested key element is not available `CRYPTO_E_KEY_READ_FAIL`: Request failed because read access was denied `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Retrieves the key element bytes from a specific key element of the key identified by the keyId and stores the key element in the memory location pointed by the key pointer.	
Available via	Csm.h	

⌋

[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.

8.3.8.4 Key Copying Interface

[SWS_Csm_00969] Definition of API function Csm_KeyElementCopy ⌈

Service Name	Csm_KeyElementCopy	
Syntax	`Std_ReturnType Csm_KeyElementCopy ( uint32 keyId, uint32 keyElementId, uint32 targetKeyId, 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, the service `Csm_KeyElementCopy` shall call `CryIf_KeyElementCopy` and pass on the return value.⌋

[SWS_Csm_01034] Definition of API function Csm_KeyCopy ⌈

Service Name	Csm_KeyCopy
Syntax	`Std_ReturnType Csm_KeyCopy ( uint32 keyId, 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, the service `Csm_KeyCopy` shall call `CryIf_KeyCopy` and pass on the return value.⌋

Note: Because there are no checks by Csm and CryIf intended related to key elements during `Csm_KeyCopy` (e.g.

- sizes of target key elements are greater or equal to the sizes of the corresponding source key elements
- access rights of source and target key elements are allowing a copy between different Crypto drivers via `Crypto_KeyElementGet` and `Crypto_KeyElementGet`)

inconsistent states of target keys are possible in case of an inappropriate configuration. Those inconsistencies have to be identified during development, and should be addressed by configuration.

[SWS_Csm_91025] Definition of API function Csm_KeyElementCopyPartial ⌈

Service Name	Csm_KeyElementCopyPartial

Syntax	`Std_ReturnType Csm_KeyElementCopyPartial (`
	`  uint32 keyId,`
	`  uint32 keyElementId,`
	`  uint32 keyElementSourceOffset,`
	`  uint32 keyElementTargetOffset,`
	`  uint32 keyElementCopyLength,`
	`  uint32 targetKeyId,`
	`  uint32 targetKeyElementId`
	`)`

Service ID [hex]	0x79

▽

△

Sync/Async	Synchronous	
Reentrancy	Reentrant, but not for the same keyId	
Parameters (in)	keyId	Holds the identifier of the key whose key element shall be the source element for copy operation.
	keyElementId	Holds the identifier of the key element which shall be the source for the copy operation.
	keyElementSourceOffset	This is the offset of the source key element indicating the start index of the copy operation.
	keyElementTargetOffset	This is the offset of the destination key element indicating the start index of the copy operation.
	keyElementCopyLength	Specifies the number of bytes that shall be copied.
	targetKeyId	Holds the identifier of the key whose key element shall be the destination element.
	targetKeyElementId	Holds the identifier of the key element which shall be the destination for the copy operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, Crypto Driver Object is busy `CRYPTO_E_KEY_NOT_AVAILABLE`: Request failed, the requested key element is not available `CRYPTO_E_KEY_READ_FAIL`: Request failed, not allowed to extract key element `CRYPTO_E_KEY_WRITE_FAIL`: Request failed, not allowed to write key element `CRYPTO_E_KEY_SIZE_MISMATCH`: Request failed, key element sizes are not compatible `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Copies a key element to another key element in the same crypto driver. The keyElementSource Offset 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 `keyElementTarget-Offset` properly.

[SWS_Csm_91019] ⌈If no errors are detected by Csm shall call `CryIf_Key-ElementCopyPartial` 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` + `key-ElementCopyLength`), then the source data shall be copied into the target key el-

ement and the length shall be set to (keyElementTargetOffset + keyElement-CopyLength).⌋

[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.⌋

8.3.8.5 Key Generation Interface

The key generation interface is used to generate a key into the key element CRYPTO_KE_KEYGENERATE_KEY according to the algorithm defined in the key element CRYPTO_KE_KEYGENERATE_ALGORITHM.

The key will be generated from the random value that is located in the key element CRYPTO_KE_KEYGENERATE_SEED.

The random value can be generated, for example, with the function Csm_Random-Generate and must be stored in CRYPTO_KE_KEYGENERATE_SEED before the key generation is triggered.

It is important to check the quality of the randomness and its entropy of the seed, which depends on the used hardware, and software of a stack. The randomness has a major impact on the quality of the generated key material.

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

[SWS_Csm_01051] Definition of API function Csm_RandomSeed ⌈

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

▽

△

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
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.⌋

[SWS_Csm_00955] Definition of API function Csm_KeyGenerate ⌈

Service Name	Csm_KeyGenerate	
Syntax	`Std_ReturnType Csm_KeyGenerate ( uint32 keyId )`	
Service ID [hex]	0x6a	
Sync/Async	Synchronous	
Reentrancy	Reentrant but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key for which a new material shall be 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	

⌋

[SWS_Csm_01005] ⌈If no errors are detected by Csm, the service Csm_KeyGenerate shall call CryIf_KeyGenerate.⌋

8.3.8.6 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_KeyDerive interface.

[SWS_Csm_00956] Definition of API function Csm_KeyDerive ⌈

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

⌋

[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`.⌋

8.3.8.7 Key Exchange Interface

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

The functions `Csm_KeyExchangeCalcPubVal` / `Csm_JobKeyExchangeCalcPubVal` and `Csm_KeyExchangeCalcSecret` / `Csm_JobKeyExchangeCalcSecret` are used to support Diffie-Hellman (DH) key exchange.

This allows two partners, Alice and Bob, to generate private and public key material, to exchange public parts so that both parties can generate at the end a common shared

secret. This shared secret can further be used, e.g. for symmetric data operation such as data encryption or MAC generation.

The public and private key material can either be based on prime based large number as it is used with RSA or on elliptic curve (so-called elliptic-curve diffie-hellman).

The CSM key exchange functions require a key with key elements according to [SWS_Csm_01022], in the line of Crypto Service "Key Exchange". The key elements CRYPTO_KE_KEYEXCHANGE_BASE, CRYPTO_KE_KEYEXCHANGE_PRIVKEY and CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY are used to hold the public/private key material.

These values can either be pre-defined and set by Csm_KeyElementSet followed by Csm_KeySetValid or generated. For example, these key values can be generated by Csm_KeyGenerate and then copied with Csm_KeyElementCopy to the corresponding key elements, followed by a call to Csm_KeySetValid.

In a first step, Alice will call Csm_KeyExchangeCalcPubVal / Csm_JobKeyExchangeCalcPubVal and send the results to Bob (exchanged data may need to be signed and/or encrypted depending on the protocol).

It should be noted, that if Csm_KeyExchangeCalcPubVal is called but no valid key material exists (key is not valid or essential key elements have length=0), the function shall generate the necessary key material and continue as normal.

If needed, Bob will put received key material from Alice into the corresponding key elements. He will also call Csm_KeyExchangeCalcPubVal to generate his shared value that needs to be sent to Alice. Afterwards, Bob can call Csm_KeyExchangeCalcSecret to generate the common secret. This value will be placed into the key element CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE.

When Alice receives the public value from Bob, it will call Csm_KeyExchangeCalcSecret and provides the value from Bob in the parameter of the function. The common shared secret will be generated by this function into the key element CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE.

Depending on the algorithm, Bob needs to send key material to Alice to allow her to generate the common shared secret.

Additional elliptic curve parameter can be specified with the additional key element CRYPTO_KE_KEYEXCHANGE_CURVETYPE.

The other key elements have the following meaning:

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

[SWS_Csm_00966] Definition of API function Csm_KeyExchangeCalcPubVal ⌈

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

⌋

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

[SWS_Csm_00967] Definition of API function Csm_KeyExchangeCalcSecret ⌈

Service Name	Csm_KeyExchangeCalcSecret	
Syntax	`Std_ReturnType Csm_KeyExchangeCalcSecret ( uint32 keyId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength )`	
Service ID [hex]	0x6d	
Sync/Async	Synchronous	
Reentrancy	Reentrant but not for same keyId	
Parameters (in)	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
	partnerPublicValuePtr	Holds the pointer to the memory location which contains the partner's public value.
	partnerPublicValueLength	Contains the length of the partner's public value in bytes.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Calculates the shared secret key for the key exchange with the key material of the key identified by the keyId and the partner public key. The shared secret key is stored as a key element in the same key.	
Available via	Csm.h	

⌋

[SWS_Csm_01006] ⌈If no errors are detected by Csm, the service Csm_KeyExchangeCalcSecret shall call CryIf_KeyExchangeCalcSecret.⌋

8.3.9 Cryptographic Primitives and Schemes

[SWS_Csm_91027] Definition of API function Csm_JobKeySetValid ⌈

Service Name	Csm_JobKeySetValid	
Syntax	`Std_ReturnType Csm_JobKeySetValid ( uint32 jobId )`	
Service ID [hex]	0x7a	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
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 associated with the job identified by jobId to valid.	
Available via	Csm.h	

⌋

[SWS_Csm_91002] Definition of API function Csm_JobKeySetInvalid ⌈

Service Name	Csm_JobKeySetInvalid	
Syntax	`Std_ReturnType Csm_JobKeySetInvalid ( uint32 jobId )`	
Service ID [hex]	0x84	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.

▽

△

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 status to invalid. The key cannot be used any longer for cryptographic operations until it has been set to valid state again.	
Available via	Csm.h	

⌋

[SWS_Csm_91028] Definition of API function Csm_JobRandomSeed ⌈

Service Name	Csm_JobRandomSeed	
Syntax	`Std_ReturnType Csm_JobRandomSeed ( uint32 jobId, const uint8* seedPtr, uint32 seedLength )`	
Service ID [hex]	0x7b	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	seedPtr	Holds a pointer to the memory location which contains the data to feed the seed.
	seedLength	Contains the length of the seed in bytes.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Provides a new seed for the key associated with the job identified by jobId that is used for the associated random number generator.	
Available via	Csm.h	

⌋

[SWS_Csm_91029] Definition of API function Csm_JobKeyGenerate ⌈

Service Name	Csm_JobKeyGenerate	
Syntax	`Std_ReturnType Csm_JobKeyGenerate ( uint32 jobId )`	
Service ID [hex]	0x7c	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
Parameters (inout)	None	

▽

△

Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Generates new key material and stores it in the key associated with the job identified by jobId.	
Available via	Csm.h	

⌋

[SWS_Csm_91030] Definition of API function Csm_JobKeyDerive ⌈

Service Name	Csm_JobKeyDerive	
Syntax	`Std_ReturnType Csm_JobKeyDerive ( uint32 jobId, uint32 targetKeyId )`	
Service ID [hex]	0x7d	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	targetKeyId	Holds the identifier of the key which is used to store the derived key.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Derives a new key by using the key elements in the given key associated with the job identified by jobId. 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	

⌋

[SWS_Csm_91031] Definition of API function Csm_JobKeyExchangeCalcPubVal

⌈

Service Name	Csm_JobKeyExchangeCalcPubVal	
Syntax	`Std_ReturnType Csm_JobKeyExchangeCalcPubVal ( uint32 jobId, uint8* publicValuePtr, uint32* publicValueLengthPtr )`	
Service ID [hex]	0x7e	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
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 public ValuePtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out)	publicValuePtr	Contains the pointer to the data where the public value shall be stored.
Return value	Std_ReturnType	`E_OK`: Request successful `E_NOT_OK`: Request failed `CRYPTO_E_BUSY`: Request failed, service is still busy `CRYPTO_E_KEY_NOT_VALID`: Request failed, the key's state is "invalid" `CRYPTO_E_KEY_EMPTY`: Request failed because of uninitialized source key element
Description	Calculates the public value of the current user for the key exchange and stores the public key in the memory location pointed by the public value pointer.	
Available via	Csm.h	

⌋

[SWS_Csm_91032] Definition of API function Csm_JobKeyExchangeCalcSecret

⌈

Service Name	Csm_JobKeyExchangeCalcSecret	
Syntax	`Std_ReturnType Csm_JobKeyExchangeCalcSecret ( uint32 jobId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength )`	
Service ID [hex]	0x7f	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	partnerPublicValuePtr	Holds the pointer to the memory location which contains the partner's public value.
	partnerPublicValueLength	Contains the length of the partner's public value in bytes.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_NOT_VALID: Request failed, the key's state is "invalid" CRYPTO_E_KEY_EMPTY: Request failed because of uninitialized source key element
Description	Calculates the shared secret key for the key exchange using the key material of the key associated with the job identified by jobId and the partner public key. The shared secret key is stored as a key element in the same key associated with the job.	
Available via	Csm.h	

⌋

[SWS_Csm_91115] Definition of API function Csm_JobKeyWrap ⌈

Service Name	Csm_JobKeyWrap	

Syntax	`Std_ReturnType Csm_JobKeyWrap (`
	`  uint32 jobId,`
	`  uint32 sourceKeyId,`
	`  uint8* ciphertextPtr,`
	`  uint32* ciphertextLengthPtr,`
	`  uint8* authenticatorPtr,`
	`  uint32* authenticatorLengthPtr`
	`)`	

Service ID [hex]	0x89	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	sourceKeyId	Holds the identifier of the key to be wrapped.
Parameters (inout)	ciphertextLengthPtr	Contains the length in bytes of the wrapped key.
	authenticatorLengthPtr	Contains the length in bytes of the authenticator.
Parameters (out)	ciphertextPtr	References the data of the wrapped key.
	authenticatorPtr	References the data of the authenticator.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_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 CRYPTO_E_JOB_CANCELED: Request failed because the job has been canceled.
Description	Wraps the plaintext key given by sourceKeyId (in the key element with the id 1) with the key wrapping key associated with the jobId. The wrapped key will be written to ciphertextPtr. If the algorithm does provide an authenticator this will be written to authenticatorPtr. If the algorithm has no authenticator or the algorithm has an authenticator and no authentication shall be done, authenticatorLengthPtr shall be set to zero.	
Available via	Csm.h	

⌋

[SWS_Csm_91111] Definition of API function Csm_JobKeyUnwrap ⌈

Service Name	Csm_JobKeyUnwrap	
Syntax	`Std_ReturnType Csm_JobKeyUnwrap ( uint32 jobId, uint32 targetKeyId, const uint8* ciphertextPtr, uint32 ciphertextLength, const uint8* authenticatorPtr, uint32 authenticatorLength, Crypto_VerifyResultType* verifyPtr )`	
Service ID [hex]	0x8A	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
	targetKeyId	Holds the identifier of the slot where the plaintext key shall be written.
	ciphertextPtr	References the data of the wrapped key.
	ciphertextLength	Contains the length in bytes of the wrapped key.
	authenticatorPtr	References the data of the authenticator.
	authenticatorLength	Contains the length in bytes of the authenticator.
Parameters (inout)	None	
Parameters (out)	verifyPtr	Contains the pointer to the result of the verification.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_KEY_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_JOB_CANCELED: Request failed because the job has been canceled.
Description	Unwraps the wrapped key given by the ciphertextPtr with the key wrapping key associated with the jobId. The unwrapped key will be written to targetKeyId in the element with the Id 1. If an authentication shall be done, the authenticator shall be referenced with authenticatorPtr. If the algorithm has no authenticator or the algorithm has an authenticator and no authentication shall be done, authenticatorLengthPtr shall be set to zero. If the algorithm has no authentication, verifyPtr will be set to CRYPTO_E_VER_OK.	
Available via	Csm.h	

⌋

8.3.10 Context Save and Restore

[SWS_Csm_91063] Definition of API function Csm_SaveContextJob ⌈

Service Name	Csm_SaveContextJob
Syntax	`Std_ReturnType Csm_SaveContextJob ( uint32 jobId, uint8* contextBufferPtr, uint32* contextBufferLengthPtr )`

▽

△

Service ID [hex]	0x86	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job using the CSM service.
Parameters (inout)	contextBufferLengthPtr	Pointer to the buffer, where the length value is located. As input data it provides the maximum length of data available in context BufferPtr. As output data it provides the actual length of data located in contextBufferPtr (or 0 in case of a failure)
Parameters (out)	contextBufferPtr	Pointer to the buffer in the application where the context data shall be stored to.
Return value	Std_ReturnType	E_OK: Context data successfully provided. E_NOT_OK: Context data could not be provided; values are not valid. CRYPTO_E_BUSY: Request failed, service is still busy
Description	The Crypto Driver stores the internal context of the respective crypto operation to the context Buffer.	
Available via	Csm.h	

⌋

[SWS_Csm_91067] ⌈If Csm_SaveContextJob is called and the job is active, CSM shall put contextBufferPtr to job->action->jobOutputsPtr[0].dataPtr and contextBufferLengthPtr into job->action->jobOutputsPtr[0].dataLengthPtr, set job->action->jobOperationMode to CRYPTO_OPERATIONMODE_SAVE_CONTEXT and call CryIf_ProcessJob with the corresponding channelId of this job and a pointer to the job itself. Any return value from this function call shall be provided back to the application.⌋

[SWS_Csm_91064] Definition of API function Csm_RestoreContextJob ⌈

Service Name	Csm_RestoreContextJob	
Syntax	`Std_ReturnType Csm_RestoreContextJob ( uint32 jobId, const uint8* contextBufferPtr, uint32 contextBufferLength )`	
Service ID [hex]	0x87	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant	
Parameters (in)	jobId	Holds the identifier of the job where context shall be requested from.
	contextBufferPtr	Pointer to the buffer, where the context data are located that shall be restored.
	contextBufferLength	Provides the length of context data that are located in context BufferPtr.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Context data successfully restored. E_NOT_OK: Context data could not be restored. CRYPTO_E_BUSY: Request failed, service is still busy

▽

△

Description	The Crypto Driver extracts the context data from the contextBuffer and restores the internal state so that further crypto operation of this crypto service will continue at the exact point when the context was taken.
Available via	Csm.h

⌋

[SWS_Csm_91068] ⌈If `Csm_RestoreContextJob` is called and the job is active, CSM shall put `contextBufferPtr` into `job->action->jobInputsPtr[0].dataPtr` and `contextBufferLength` into `job->action->jobInputsPtr[0].dataLength`, set `job->action->jobOperationMode` to `CRYPTO_OPERATIONMODE_RESTORE_CONTEXT` and call `CryIf_ProcessJob` with the corresponding `channelId` of this job and a pointer to the job itself. Any return value from this function call shall be provided back to the application.⌋

8.3.11 Job Cancellation Interface

[SWS_Csm_00968] Definition of API function Csm_CancelJob ⌈

Service Name	Csm_CancelJob	
Syntax	`Std_ReturnType Csm_CancelJob ( uint32 job, Crypto_OperationModeType mode )`	
Service ID [hex]	0x6f	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	job	Holds the identifier of the job to be canceled
	mode	Not used, just for interface compatibility provided.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request successful. Job removed from Crypto Driver Object queue OR immediate cancelation of Job processing in Crypto was possible. `E_NOT_OK`: Request failed `CRYPTO_E_JOB_CANCELED`: Immediate cancelation of Job processing in Crypto not possible. The cancelation will be done at next suitable processing step and notified via a negative job's closing callback.
Description	Cancels the job processing from asynchronous or streaming jobs.	
Available via	Csm.h	

⌋

[SWS_Csm_01021] ⌈The Csm shall call `CryIf_CancelJob` and pass on the return value.⌋

8.3.12 Custom Interface

8.3.12.1 Custom Service Interface

Security related services that are not supported by the existing CSM service interfaces can be realized by the custom service. The purpose of this service is not specified and can be customized by a vendor. It is recommended to align the settings of a particular custom service in a Crypto Driver profile (see SWS Crypto Driver [3] Chapter 7.3.7 *"Crypto Profiles"*).

[SWS_Csm_91107] Definition of API function Csm_CustomService ⌈

Service Name	Csm_CustomService	

Syntax	`Std_ReturnType Csm_CustomService (`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  uint32 targetKeyId,`
	`  const uint8* inputPtr,`
	`  uint32 inputLength,`
	`  const uint8* secondaryInputPtr,`
	`  uint32 secondaryInputLength,`
	`  const uint8* tertiaryInputPtr,`
	`  uint32 tertiaryInputLength,`
	`  uint8* outputPtr,`
	`  uint32* outputLengthPtr,`
	`  uint8* secondaryOutputPtr,`
	`  uint32* secondaryOutputLengthPtr,`
	`  Crypto_VerifyResultType* verifyPtr`
	`)`	

Service ID [hex]	0x13	
Sync/Async	Depends on configuration	
Reentrancy	Reentrant for different jobId	
Parameters (in)	jobId	ID of the job which is dispatched for custom processing
	mode	Indicates which operation mode(s) to perform for the custom Job
	targetKeyId	Holds the target key id for the custom job
	inputPtr	Pointer to the input data.
	inputLength	Contains the input length in bytes.
	secondaryInputPtr	Pointer to the secondary input data.
	secondaryInputLength	Contains the secondary input length in bytes.
	tertiaryInputPtr	Pointer to the tertiary input data.
	tertiaryInputLength	Contains the tertiary input length in bytes.
Parameters (inout)	None	
Parameters (out)	outputPtr	Pointer to the output data.
	outputLengthPtr	Contains the output length in bytes.
	secondaryOutputPtr	Pointer to the secondary output data.
	secondaryOutputLength Ptr	Contains the secondary output length in bytes.
	verifyPtr	Pointer to the verification result

▽

△

Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, service is still busy CRYPTO_E_CUSTOM_ERROR: Custom processing failed.
Description	Dispatches security related jobs for custom execution in a secure environment	
Available via	Csm.h	

⌋

8.3.12.2 Custom Sync Interface

[SWS_Csm_91108] Definition of API function Csm_CustomSync ⌈

Service Name	Csm_CustomSync	

Syntax	`Std_ReturnType Csm_CustomSync (`
	`  uint32 dispatchId,`
	`  uint32 keyId,`
	`  uint32 keyElementId,`
	`  uint32 targetKeyId,`
	`  uint32 targetKeyElementId,`
	`  const uint8* inputPtr,`
	`  uint32 inputLength,`
	`  uint8* outputPtr,`
	`  uint32* outputLengthPtr,`
	`  uint8* secondaryOutputPtr,`
	`  uint32* secondaryOutputLengthPtr`
	`)`	

Service ID [hex]	0x88	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	dispatchId	unique id to identify the request
	keyId	key Id of the key the certificate is stored
	keyElementId	key element id
	targetKeyId	Holds the target key id
	targetKeyElementId	–
	inputPtr	Pointer to the input data.
	inputLength	Contains the input length in bytes.
Parameters (inout)	None	
Parameters (out)	outputPtr	Pointer to the output data.
	outputLengthPtr	Contains the output length in bytes.
	secondaryOutputPtr	Pointer to the secondary output data.
	secondaryOutputLength Ptr	Contains the secondary output length in bytes.
Return value	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: The service request failed because the service is still busy CRYPTO_E_CUSTOM_ERROR: Custom processing failed
Description	Requests the execution of a function that is specified by the given dispatch id.	
Available via	Csm.h	

⌋

[SWS_Csm_91110] ⌈If no errors are detected by Csm, the service `Csm_CustomSync` shall call `CryIf_CustomSync`.⌋

8.3.13 Callback notifications

This is a list of functions provided for other modules.

[SWS_Csm_00970] Definition of API function Csm_CallbackNotification

Upstream requirements: SRS_BSW_00359, SRS_BSW_00360

⌈

Service Name	Csm_CallbackNotification	
Syntax	`void Csm_CallbackNotification ( Crypto_JobType* job, Crypto_ResultType result )`	
Service ID [hex]	0x70	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	job	Holds a pointer to the job, which has finished.
	result	Contains the result of the cryptographic operation.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notifies the CSM that a job has finished. This function is used by the underlying layer (CRYIF).	
Available via	Csm.h	

⌋

[SWS_Csm_01044] ⌈If the `CRYPTO_OPERATIONMODE_FINISH` bit is set in `job->action->jobOperationMode`, the `Csm_CallbackNotification` shall call the configured callback function.⌋

[SWS_Csm_91017] ⌈If the `CRYPTO_OPERATIONMODE_FINISH` bit is set in `job->action->jobOperationMode` and `CsmProcessingMode` is set to `CRYPTO_PROCESSING_ASYNC` and `CsmJobInterfaceUsePort` is set to `CRYPTO_USE_PORT_OPTIMIZED`, the CSM shall trigger CallbackNotification service.⌋

8.4 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

[SWS_Csm_00479] Definition of scheduled function Csm_MainFunction_<shortName>

Upstream requirements: SRS_BSW_00373, SRS_BSW_00432

⌈

Service Name	Csm_MainFunction_<shortName>
Syntax	`void Csm_MainFunction_<shortName> ( void )`
Service ID [hex]	0x01
Description	API to be called cyclically. Hereby <shortName> is the short name of the `CsmMainFunction` configuration container in the ECU configuration.
Available via	SchM_Csm.h

⌋

8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[SWS_Csm_91100] Definition of mandatory interfaces required by module Csm

⌈

API Function	*Header File*	*Description*
CryIf_CancelJob	CryIf.h	This interface dispatches the job cancellation function to the configured crypto driver object.
CryIf_KeyCopy	CryIf.h	This function shall copy all key elements from the source key to a target key.
CryIf_KeyElementCopy	CryIf.h	This function shall copy a key elements from one key to a target key.
CryIf_KeyElementCopyPartial	CryIf.h	Copies a key element to another key element. The keyElementOffsets and keyElementCopyLength allows to copy just parts of the source key element into the destination key element.
CryIf_KeyElementGet	CryIf.h	This function shall dispatch the get key element function to the configured crypto driver object.
CryIf_KeyElementSet	CryIf.h	This function shall dispatch the set key element function to the configured crypto driver object.
CryIf_KeyExchangeCalcSecret	CryIf.h	This function shall dispatch the key exchange common shared secret calculation function to the configured crypto driver object.
CryIf_KeyGenerate	CryIf.h	This function shall dispatch the key generate function to the configured crypto driver object.

▽

△

API Function	Header File	Description
CryIf_KeySetValid	CryIf.h	This function shall dispatch the set key valid function to the configured crypto driver object.
CryIf_ProcessJob	CryIf.h	This interface dispatches the received jobs to the configured crypto driver object.
CryIf_RandomSeed	CryIf.h	This function shall dispatch the random seed function to the configured crypto driver object.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

[SWS_Csm_91101] Definition of optional interfaces requested by module Csm ⌈

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.

⌋

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[SWS_Csm_00971] Definition of configurable interface <Csm_ApplicationCallbackNotification>

Upstream requirements: SRS_BSW_00359, SRS_BSW_00360

⌈

Service Name	<Csm_ApplicationCallbackNotification>	
Syntax	`void <Csm_ApplicationCallbackNotification> ( uint32 jobId, Crypto_ResultType result )`	
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. The function name itself is derived from "{CsmJob/CsmJobPrimitiveCallbackRef}/CsmCallback Func".
Available via	Csm.h

⌋

[SWS_Csm_01090] ⌈<Csm_ApplicationCallbackNotification> shall be called once at the end when an asynchronous job's call has been finished, i.e. the given operation mode has been completely processed, the job has been aborted due to an error or the the job has been cancelled. Thus, if a job's call processed multiple operation modes, i.e. CRYPTO_OPERATIONMODE_STREAMSTART or CRYPTO_OPERATIONMODE_SINGLECALL, <Csm_ApplicationCallbackNotification> is called only once.⌋

[SWS_Csm_01095] ⌈The CSM shall call <Csm_Application-CallbackNotification> if the following condition is met: `({ecuc(Csm/CsmJobs/CsmJob.CsmProcessingMode)} == CRYPTO_PROCESSING_ASYN) && (CsmJob/CsmJobInterfaceUsePort == CRYPTO_USE_FNC) && (CsmJob/CsmJobPrimitiveCallbackRef != 0)`

⌋

8.6 Service Interfaces

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.

8.6.1 Client-Server-Interfaces

[SWS_Csm_01905] Definition of ClientServerInterface CsmKeyManagement_{Key} ⌈

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	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	6	CRYPTO_E_KEY_READ_FAIL	The service request failed because read access was denied.
	7	CRYPTO_E_KEY_WRITE_FAIL	The service request failed because write access was denied.
	8	CRYPTO_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	KeyCopy	
Comment	This function shall copy all key elements from the source key to a target key.	
Relates to	Csm_KeyCopy	
Variation	–	
Parameters	targetKeyId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key whose key element shall be the destination element.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_AVAILABLE CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

Operation	KeyDerive	
Comment	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.	
Relates to	Csm_KeyDerive	
Variation	–	
Parameters	targetKeyId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key which is used to store the derived key.
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY

Operation	KeyElementCopy	
Comment	This function shall copy a key elements from one key to a target key	
Relates to	Csm_KeyElementCopy	
Variation	–	
Parameters	keyElementId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key element which shall be the source for the copy operation.
	Variation	–
	targetKeyId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key whose key element shall be the destination element.
	Variation	–
	targetKeyElementId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key element which shall be the destination for the copy operation.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_AVAILABLE CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

Operation	KeyElementCopyPartial	
Comment	This function shall copy parts of a a key elements from one key to parts of a target key element of a target key.	
Relates to	Csm_KeyElementCopyPartial	
Variation	–	
Parameters	keyElementId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key element which shall be the source for the copy operation.
	Variation	–
	keyElementSourceOffset	

▽

△

	Type	uint32
	Direction	IN
	Comment	This is the offset of the source key element indicating the start index of the copy operation.
	Variation	–
	keyElementTargetOffset	
	Type	uint32
	Direction	IN
	Comment	This is the offset of the destination key element indicating the start index of the copy operation.
	Variation	–
	keyElementCopyLength	
	Type	uint32
	Direction	IN
	Comment	Specifies the number of bytes that shall be copied.
	Variation	–
	targetKeyId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key whose key element shall be the destination element.
	Variation	–
	targetKeyElementId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key element which shall be the destination for the copy operation.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_AVAILABLE CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

Operation	KeyElementGet	
Comment	Retrieves the key element bytes from a specific key element of the key and stores the key element in the provided buffer.	
Relates to	Csm_KeyElementGet	
Variation	–	
Parameters	keyElementId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key element to be read.
	Variation	–
	keyElement	
	Type	Csm_KeyDataType_{Crypto}

▽

△

	Direction	OUT
	Comment	Holds the data to the key element bytes to be written.
	Variation	–
	keyElementLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the number of key element bytes.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_NOT_AVAILABLE CRYPTO_E_KEY_EMPTY	

Operation	KeyElementSet	
Comment	Sets the given key element bytes to the key.	
Relates to	Csm_KeyElementSet	
Variation	–	
Parameters	keyElementId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key element to be written.
	Variation	–
	keyElement	
	Type	Csm_KeyDataType_{Crypto}
	Direction	IN
	Comment	Holds the data to the key element bytes to be processed.
	Variation	–
	keyElementLength	
	Type	uint32
	Direction	IN
	Comment	Contains the number of key element bytes.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_AVAILABLE CRYPTO_E_KEY_SIZE_MISMATCH	

Operation	KeyExchangeCalcPubVal	
Comment	Calculates the public value of the current user for the key exchange and stores the public key in the provided buffer	
Relates to	Csm_KeyExchangeCalcPubVal	
Variation	–	
Parameters	publicValue	
	Type	Csm_KeyDataType_{Crypto}
	Direction	OUT

▽

△

	Comment	Contains the pointer to the memory location where the public value shall be stored.
	Variation	–
	publicValueLength	
	Type	uint32
	Direction	INOUT
	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.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY	

Operation	KeyExchangeCalcSecret	
Comment	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.	
Relates to	Csm_KeyExchangeCalcSecret	
Variation	–	
Parameters	partnerPublicValue	
	Type	Csm_KeyDataType_{Crypto}
	Direction	IN
	Comment	Holds the pointer to the memory location containing the partner's public value
	Variation	–
	partnerPublicValueLength	
	Type	uint32
	Direction	IN
	Comment	Contains the number of bytes of the partner pulic value
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY	

Operation	KeyGenerate
Comment	Generates new key material and store it in the key identified by keyId.
Relates to	Csm_KeyGenerate
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY

Operation	KeyGetStatus
Comment	Returns the key state of the key identified by keyId.
Relates to	Csm_KeyGetStatus

▽

△

Variation	–	
Parameters	keyStatusPtr	
	Type	Crypto_KeyStatusType
	Direction	OUT
	Comment	Holds the status of the key referenced by the port.
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	KeySetInvalid
Comment	Operation to set the status of the key to invalid.
Relates to	Csm_KeySetInvalid
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY

Operation	KeySetValid
Comment	Sets the given key element bytes to the key.
Relates to	Csm_KeySetValid
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY

Operation	RandomSeed	
Comment	Feeds the key element CRYPTO_KE_RANDOM_SEED with a random seed.	
Relates to	Csm_RandomSeed	
Variation	–	
Parameters	seed	
	Type	Csm_KeyDataType_{Crypto}
	Direction	IN
	Comment	Holds the data which shall be used for the random seed initialization.
	Variation	–
	seedLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length of the seed in bytes.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_00946] Definition of ClientServerInterface CsmHash_{Primitive} ⌈

Name	CsmHash_{Primitive}		
Comment	Synchronous processing interface to execute the hash calculation.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.

Operation	Hash	
Comment	Streaming approach of the hash calculation.	
Relates to	Csm_Hash	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data to be hashed.
	Variation	Service = "Hash" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data to be hashed.
	Variation	–
	result	
	Type	Csm_{Service}ResultType_{Primitive}
	Direction	OUT
	Comment	Contains the data of the hash.
	Variation	Service = "Hash" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	resultLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the hash.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY	

⌋

[SWS_Csm_09000] Definition of ClientServerInterface CsmMacGenerate_{Primitive} ⌈

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

▽

△

Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	MacGenerate	
Comment	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	
Relates to	Csm_MacGenerate	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data from which a MAC shall be generated of.
	Variation	Service = "MacGenerate" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
	Variation	–
	mac	
	Type	Csm_{Service}ResultType_{Primitive}
	Direction	OUT
	Comment	Contains the data of the MAC.
	Variation	Service = "MacGenerate" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	macLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the MAC.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_00936] Definition of ClientServerInterface CsmMacVerify_{Primitive}
⌈

Name	CsmMacVerify_{Primitive}		
Comment	Synchronous processing interface to execute the MAC verification.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	MacVerify	
Comment	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	
Relates to	Csm_MacVerify	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data from which a MAC shall be generated of.
	Variation	Service = "MacVerify" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data for which MAC shall be verified.
	Variation	–
	mac	
	Type	Csm_{Service}CompareType_{Primitive}
	Direction	IN
	Comment	Contains the MAC to be verified.
	Variation	Service = "MacVerify" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	macLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in BITS of the MAC to be verified.
	Variation	–
	verify	
	Type	Crypto_VerifyResultType
	Direction	OUT
	Comment	Contains the verification result.
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY

⌋

[SWS_Csm_00947] Definition of ClientServerInterface CsmEncrypt_{Primitive} ⌈

Name	CsmEncrypt_{Primitive}		
Comment	Synchronous processing interface to execute the encryption.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	Encrypt	
Comment	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.	
Relates to	Csm_Encrypt	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data to be encrypted.
	Variation	Service = "Encrypt" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data to be encrypted.
	Variation	–
	result	
	Type	Csm_{Service}ResultType_{Primitive}
	Direction	OUT
	Comment	Contains the data of the cipher.
	Variation	Service = "Encrypt" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	resultLength	

▽

△

	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the cipher.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_01906] Definition of ClientServerInterface CsmDecrypt_{Primitive} ⌈

Name	CsmDecrypt_{Primitive}		
Comment	Synchronous processing interface to execute the decryption.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	Decrypt	
Comment	Streaming approach of the decryption.	
Relates to	Csm_Decrypt	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data to be decrypted.
	Variation	Service = "Decrypt" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data to be decrypted.
	Variation	–
	result	
	Type	Csm_{Service}ResultType_{Primitive}
	Direction	OUT

▽

△

	Comment	Contains the data of the decrypted plaintext.
	Variation	Service = "Decrypt" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	resultLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the decrypted plaintext.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_01910] Definition of ClientServerInterface CsmAEADEncrypt_{Primitive} ⌈

Name	CsmAEADEncrypt_{Primitive}		
Comment	Synchronous processing interface to execute the AEAD encryption.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	AEADEncrypt	
Comment	Streaming approach of the AEAD encryption.	
Relates to	Csm_AEADEncrypt	
Variation	–	
Parameters	plaintext	
	Type	Csm_AEADEncryptPlaintextType_{Primitive}
	Direction	IN
	Comment	Contains the plaintext to be encrypted with AEAD.
	Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	plaintextLength	
	Type	uint32
	Direction	IN
	Comment	This element Contains the length in bytes of the plaintext to be encrypted with AEAD.

▽

△

	Variation	–
	associatedData	
	Type	Csm_{Service}AssociatedDataType_{Primitive}
	Direction	IN
	Comment	Contains the data of the header (that is not part of the encryption but authentication).
	Variation	Service = "AEADEncrypt" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	associatedDataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data of the header.
	Variation	–
	ciphertext	
	Type	Csm_AEADEncryptCiphertextType_{Primitive}
	Direction	OUT
	Comment	Contains the data of the AEAD cipher.
	Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	ciphertextLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the data of the AEAD cipher.
	Variation	–
	tag	
	Type	Csm_AEADEncryptTagType_{Primitive}
	Direction	OUT
	Comment	Contains the data of the Tag.
	Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	tagLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the data of the Tag.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_01915] Definition of ClientServerInterface CsmAEADDecrypt_{Primitive} ⌈

Name	CsmAEADDecrypt_{Primitive}		
Comment	Synchronous processing interface to execute the AEAD decryption.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	AEADDecrypt	
Comment	Streaming approach of the AEAD decryption.	
Relates to	Csm_AEADDecrypt	
Variation	–	
Parameters	ciphertext	
	Type	Csm_AEADDecryptCiphertextType_{Primitive}
	Direction	IN
	Comment	Contains the ciphertext to be decrypted with AEAD.
	Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	ciphertextLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the ciphertext to be decrypted with AEAD.
	Variation	–
	associatedData	
	Type	Csm_{Service}AssociatedDataType_{Primitive}
	Direction	IN
	Comment	Contains the data of the header (that is not part of the encryption but authentication) .
	Variation	Service = "AEADDecrypt" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	associatedDataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data of the header.
	Variation	–
	tag	
	Type	Csm_AEADDecryptTagType_{Primitive}
	Direction	IN
	Comment	Contains the data of the Tag.
	Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	tagLength	

▽

△

	Type	uint32
	Direction	IN
	Comment	Contains the length in BITS of the data of the Tag.
	Variation	–
	plaintext	
	Type	Csm_AEADDecryptPlaintextType_{Primitive}
	Direction	OUT
	Comment	Contains the data of the decrypted AEAD plaintext.
	Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	plaintextLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the data of the decrypted AEAD plaintext.
	Variation	–
	verify	
	Type	Crypto_VerifyResultType
	Direction	OUT
	Comment	Contains the verification result.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_00903] Definition of ClientServerInterface CsmSignatureGenerate_{Primitive} ⌈

Name	CsmSignatureGenerate_{Primitive}		
Comment	Synchronous processing interface to generate a signature.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	SignatureGenerate	
Comment	Streaming approach of the signature generation.	
Relates to	Csm_SignatureGenerate	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data from which the signature shall be generated.
	Variation	Service = "SignatureGenerate" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data from which the signature shall be generated.
	Variation	–
	signature	
	Type	Csm_{Service}ResultType_{Primitive}
	Direction	OUT
	Comment	Contains the signature.
	Variation	Service = "SignatureGenerate" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	signatureLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the signature.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_00943] Definition of ClientServerInterface CsmSignatureVerify_{Primitive} ⌈

Name	CsmSignatureVerify_{Primitive}		
Comment	Synchronous processing interface to execute the signature verification.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.

▽

△

	10	CRYPTO_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.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	SignatureVerify	
Comment	Interface to verify a signature.	
Relates to	Csm_SignatureVerify	
Variation	–	
Parameters	data	
	Type	Csm_{Service}DataType_{Primitive}
	Direction	IN
	Comment	Contains the data for whichs signature shall be verified.
	Variation	Service = "SignatureVerify" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data for which signature shall be verified.
	Variation	–
	signature	
	Type	Csm_{Service}CompareType_{Primitive}
	Direction	IN
	Comment	Contains the signature to be verified.
	Variation	Service = "SignatureVerify" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	signatureLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the signature to be verified.
	Variation	–
	verify	
	Type	Crypto_VerifyResultType
	Direction	OUT
	Comment	Contains the verification result.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_00902] Definition of ClientServerInterface CsmRandomGenerate_{Primitive} ⌈

Name	CsmRandomGenerate_{Primitive}		
Comment	Synchronous processing interface to execute the random number generation.		
IsService	true		
Variation	Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	4	CRYPTO_E_ENTROPY_EXHAUSTED	Request failed, entropy of random number generator is exhausted.

Operation	RandomGenerate	
Comment	Synchronous processing interface to execute the random number generation.	
Relates to	Csm_RandomGenerate	
Variation	–	
Parameters	result	
	Type	Csm_{Service}ResultType_{Primitive}
	Direction	OUT
	Comment	Contains the random number
	Variation	Service = "RandomGenerate" Primitive = {ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
	resultLength	
	Type	uint32
	Direction	INOUT
	Comment	Contains the length in bytes of the data of random number.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_ENTROPY_EXHAUSTED	

⌋

8.6.2 Client-Server Interfaces (DATA_REFERENCES)

[SWS_Csm_91051] Definition of ClientServerInterface CsmHash ⌈

Name	CsmHash		
Comment	Asynchronous processing interface to execute the hash calculation.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	Hash	
Comment	Utilize the random seed service.	
Relates to	Csm_Hash	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data to be hashed.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data to be hashed.
	Variation	–
	resultPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the hash.
	Variation	–
	resultLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the hash.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY	

⌋

[SWS_Csm_91052] Definition of ClientServerInterface CsmMacGenerate ⌈

Name	CsmMacGenerate		
Comment	Asynchronous processing interface to execute the MAC generation.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.

▽

△

	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	MacGenerate	
Comment	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	
Relates to	Csm_MacGenerate	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data from which a MAC shall be generated of.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
	Variation	–
	macPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the MAC.
	Variation	–
	macLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the MAC.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91053] Definition of ClientServerInterface CsmMacVerify ⌈

Name	CsmMacVerify		
Comment	Asynchronous processing interface to execute the MAC verification.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	MacVerify	
Comment	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	
Relates to	Csm_MacVerify	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data from which a MAC shall be generated of.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data for which MAC shall be verified.
	Variation	–
	macPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the MAC to be verified.
	Variation	–
	macLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in BITS of the MAC to be verified.



△

	Variation	–
	verifyPtr	
	Type	Csm_VerifyResultPtr
	Direction	IN
	Comment	Contains the verification result.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91054] Definition of ClientServerInterface CsmEncrypt ⌈

Name	CsmEncrypt		
Comment	Asynchronous processing interface to execute the encryption.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	Encrypt	
Comment	Encrypts the given data and stores the ciphertext in the memory location pointed by the result pointer.	
Relates to	Csm_Encrypt	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN

▽

△

	Comment	References the data to be encrypted.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data to be encrypted.
	Variation	–
	resultPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the cipher.
	Variation	–
	resultLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the cipher.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91055] Definition of ClientServerInterface CsmDecrypt ⌈

Name	CsmDecrypt		
Comment	Asynchronous processing interface to execute the decryption.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	Decrypt	
Comment	Decrypts the given data and stores the plaintext in the memory location pointed by the result Buffer pointer.	
Relates to	Csm_Decrypt	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data to be decrypted.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data to be decrypted.
	Variation	–
	resultPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the decrypted plaintext.
	Variation	–
	resultLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the decrypted plaintext.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91056] Definition of ClientServerInterface CsmAEADEncrypt ⌈

Name	CsmAEADEncrypt		
Comment	Asynchronous processing interface to execute the AEAD encryption.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.

▽

△

	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	AEADEncrypt	
Comment	Streaming approach of the AEAD encryption.	
Relates to	Csm_AEADEncrypt	
Variation	–	
Parameters	plaintextPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the plaintext to be encrypted with AEAD.
	Variation	–
	plaintextLength	
	Type	uint32
	Direction	IN
	Comment	This element Contains the length in bytes of the plaintext to be encrypted with AEAD.
	Variation	–
	associatedDataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data of the header (that is not part of the encryption but authentication).
	Variation	–
	associatedDataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data of the header.
	Variation	–
	ciphertextPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the AEAD cipher.
	Variation	–
	ciphertextLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the data of the AEAD cipher.
	Variation	–
	tagPtr	
	Type	VoidPtr
	Direction	IN

▽

△

	Comment	References the data of the Tag.
	Variation	–
	tagLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the data of the Tag.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

⌋

[SWS_Csm_91057] Definition of ClientServerInterface CsmAEADDecrypt ⌈

Name	CsmAEADDecrypt		
Comment	Asynchronous processing interface to execute the AEAD decryption.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	AEADDecrypt	
Comment	Streaming approach of the AEAD decryption.	
Relates to	Csm_AEADDecrypt	
Variation	–	
Parameters	ciphertextPtr	
	Type	ConstVoidPtr
	Direction	IN

▽

△

	Comment	References the ciphertext to be decrypted with AEAD.
	Variation	–
	ciphertextLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the ciphertext to be decrypted with AEAD.
	Variation	–
	associatedDataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data of the header (that is not part of the encryption but authentication).
	Variation	–
	associatedDataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data of the header.
	Variation	–
	tagPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data of the Tag.
	Variation	–
	tagLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in BITS of the data of the Tag.
	Variation	–
	plaintextPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the decrypted AEAD plaintext.
	Variation	–
	plaintextLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the data of the decrypted AEAD plaintext.
	Variation	–
	verifyPtr	
	Type	Csm_VerifyResultPtr
	Direction	IN
	Comment	Contains the verification result.
	Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

⌋

[SWS_Csm_91058] Definition of ClientServerInterface CsmSignatureGenerate ⌈

Name	CsmSignatureGenerate		
Comment	Asynchronous processing interface to generate a signature.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	SignatureGenerate	
Comment	Operation to generate a signature.	
Relates to	Csm_SignatureGenerate	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN

▽

△

	Comment	References the data from which the signature shall be generated.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data from which the signature shall be generated.
	Variation	–
	resultPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the signature.
	Variation	–
	resultLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the signature.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91059] Definition of ClientServerInterface CsmSignatureVerify ⌈

Name	CsmSignatureVerify		
Comment	Asynchronous processing interface to execute the signature verification.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob

▽

△

Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	SignatureVerify	
Comment	Operation to verify a signature.	
Relates to	Csm_SignatureVerify	
Variation	–	
Parameters	dataPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data for which signature shall be verified.
	Variation	–
	dataLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the data for which signature shall be verified.
	Variation	–
	comparePtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the signature to be verified.
	Variation	–
	compareLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the signature to be verified.
	Variation	–
	verifyPtr	
	Type	Csm_VerifyResultPtr
	Direction	IN
	Comment	Contains the verification result.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91060] Definition of ClientServerInterface CsmRandomGenerate ⌈

Name	CsmRandomGenerate
Comment	Asynchronous processing interface to execute the random number generation.
IsService	true

▽

△

Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	4	CRYPTO_E_ENTROPY_EXHAUSTED	Request failed, entropy of random number generator is exhausted.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	RandomGenerate	
Comment	Generates a random number and stores it in the memory location pointed by the resultBuffer pointer.	
Relates to	Csm_RandomGenerate	
Variation	–	
Parameters	resultPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the random number.
	Variation	–
	resultLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the data of random number.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_ENTROPY_EXHAUSTED	

⌋

[SWS_Csm_91109] Definition of ClientServerInterface CsmCustom ⌈

Name	CsmCustom		
Comment	Asynchronous processing interface to execute custom crypto operation.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.

▽

△

	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	14	CRYPTO_E_CUSTOM_ERROR	–

Operation	CancelJob
Comment	Cancels the Job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	CustomService	
Comment	Operation to execute custom crypto job.	
Relates to	Csm_CustomService	
Variation	–	
Parameters	targetKeyId	
	Type	uint32
	Direction	IN
	Comment	–
	Variation	–
	inputPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	–
	Variation	–
	inputLength	
	Type	uint32
	Direction	IN
	Comment	–
	Variation	–
	secondaryInputPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	–
	Variation	–
	secondaryInputLength	
	Type	uint32
	Direction	IN
	Comment	–
	Variation	–
	tertiaryInputPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	–
	Variation	–

▽

△

	tertiaryInputLength	
	Type	uint32
	Direction	IN
	Comment	–
	Variation	–
	outputPtr	
	Type	VoidPtr
	Direction	IN
	Comment	–
	Variation	–
	outputLengthPtr	
	Type	uint32*
	Direction	IN
	Comment	–
	Variation	–
	secondaryOutputPtr	
	Type	VoidPtr
	Direction	IN
	Comment	–
	Variation	–
	secondaryOutputLengthPtr	
	Type	uint32*
	Direction	IN
	Comment	–
	Variation	–
	verifyPtr	
	Type	Crypto_VerifyResultType*
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_JOB_CANCELED CRYPTO_E_CUSTOM_ERROR	

⌋

8.6.3 Client-Server-Interfaces (Key Management)

[SWS_Csm_91035] Definition of ClientServerInterface CsmJobKeySetValid ⌈

Name	CsmJobKeySetValid
Comment	Interface to set a key valid.
IsService	true

▽

△

Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeySetValid
Comment	Operation to set a key valid.
Relates to	Csm_JobKeySetValid
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY

⌋

[SWS_Csm_91003] Definition of ClientServerInterface CsmJobKeySetInvalid ⌈

Name	CsmJobKeySetInvalid		
Comment	Interface to set the status of the key to invalid.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeySetInvalid
Comment	Operation to set the status of the key to invalid.
Relates to	Csm_JobKeySetInvalid
Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY

⌋

[SWS_Csm_91036] Definition of ClientServerInterface CsmJobRandomSeed ⌈

Name	CsmJobRandomSeed		
Comment	Interface to random seed operation.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	RandomSeed	
Comment	Utilize the random seed service.	
Relates to	Csm_JobRandomSeed	
Variation	–	
Parameters	seedPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	Holds the data which shall be used for the random seed initialization.
	Variation	–
	seedLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length of the seed in bytes.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91037] Definition of ClientServerInterface CsmJobKeyGenerate ⌈

Name	CsmJobKeyGenerate		
Comment	Interface to execute key generation.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeyGenerate
Comment	Generates new key material and stores it in the key identified by keyId.
Relates to	Csm_JobKeyGenerate
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY

⌋

[SWS_Csm_91038] Definition of ClientServerInterface CsmJobKeyDerive ⌈

Name	CsmJobKeyDerive		
Comment	Interface to execute key derive.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	6	CRYPTO_E_KEY_READ_FAIL	The service request failed because read access was denied.
	7	CRYPTO_E_KEY_WRITE_FAIL	The service request failed because write access was denied.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.

▽

△

	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeyDerive	
Comment	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.	
Relates to	Csm_JobKeyDerive	
Variation	–	
Parameters	targetKeyId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key which is used to store the derived key.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91039] Definition of ClientServerInterface CsmJobKeyExchangeCalcPubVal ⌈

Name	CsmJobKeyExchangeCalcPubVal		
Comment	Interface to execute calculation of the public value for key exchange.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.

▽

△

	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeyExchangeCalcPubVal	
Comment	Calculates the public value of the current user for the key exchange and stores the public key in the provided buffer.	
Relates to	Csm_JobKeyExchangeCalcPubVal	
Variation	–	
Parameters	publicValuePtr	
	Type	VoidPtr
	Direction	IN
	Comment	Contains the pointer to the memory location where the public value shall be stored.
	Variation	–
	publicValueLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	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.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91040] Definition of ClientServerInterface CsmJobKeyExchangeCalcSecret ⌈

Name	CsmJobKeyExchangeCalcSecret		
Comment	Interface to execute calculation of shared secret for key exchange.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.

▽

△

	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeyExchangeCalcSecret	
Comment	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.	
Relates to	Csm_JobKeyExchangeCalcSecret	
Variation	–	
Parameters	partnerPublicValuePtr	
	Type	Csm_KeyDataType_{Crypto}
	Direction	IN
	Comment	Holds the pointer to the memory location containing the partner's public value.
	Variation	–
	partnerPublicValueLength	
	Type	uint32
	Direction	IN
	Comment	Contains the number of bytes of the partner public value.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91113] Definition of ClientServerInterface CsmJobKeyWrap ⌈

Name	CsmJobKeyWrap		
Comment	Interface to execute key wrap.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.

▽

△

	6	CRYPTO_E_KEY_READ_FAIL	The service request failed because read access was denied.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.
	13	CRYPTO_E_KEY_EMPTY	Request failed because of uninitialized source key element

Operation	CancelJob
Comment	–
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeyWrap	
Comment	Wraps the plaintext key given by sourceKeyId (in the key element with the id 1) with the key wrapping key associated with the jobId. The wrapped key will be written to ciphertextPtr. If the algorithm does provide an authenticator this will be written to authenticatorPtr. If the algorithm has no authenticator or the algorithm has an authenticator and no authentication shall be done, authenticatorLengthPtr shall be set to zero.	
Relates to	Csm_JobKeyWrap	
Variation	–	
Parameters	sourceKeyId	
	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the key to be wrapped.
	Variation	–
	ciphertextPtr	
	Type	VoidPtr
	Direction	IN
	Comment	References the data of the wrapped key.
	Variation	–
	ciphertextLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the wrapped key.
	Variation	–
	authenticatorPtr	
	Type	VoidPtr
	Direction	IN
	Comment	Contains the pointer to the data of the authenticator.
	Variation	–
	authenticatorLengthPtr	

▽

△

	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the authenticator.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_READ_FAIL CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_JOB_CANCELED CRYPTO_E_KEY_EMPTY	

⌋

[SWS_Csm_91114] Definition of ClientServerInterface CsmJobKeyUnwrap ⌈

Name	CsmJobKeyUnwrap		
Comment	Interface to execute key unwrap.		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.
	7	CRYPTO_E_KEY_WRITE_FAIL	The service request failed because write access was denied.
	9	CRYPTO_E_KEY_NOT_VALID	Request failed, the key is not valid.
	10	CRYPTO_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.
	12	CRYPTO_E_JOB_CANCELED	Request failed because the job has been canceled.

Operation	CancelJob
Comment	Cancels the job.
Relates to	Csm_CancelJob
Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_JOB_CANCELED

Operation	KeyUnwrap
Comment	Unwraps the wrapped key given by the ciphertextPtr with the key wrapping key associated with the jobId. The unwrapped key will be written to targetKeyId in the element with the Id 1. If an authentication shall be done, the authenticator shall be referenced with authenticatorPtr. If the algorithm has no authenticator or the algorithm has an authenticator and no authentication shall be done, authenticatorLengthPtr shall be set to zero. If the algorithm has no authentication, verifyPtr will be set to CRYPTO_E_VER_OK.
Relates to	Csm_JobKeyUnwrap
Variation	–
Parameters	targetKeyId

▽

△

	Type	uint32
	Direction	IN
	Comment	Holds the identifier of the slot where the plaintext key shall be written.
	Variation	–
	ciphertextPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	References the data of the wrapped key.
	Variation	–
	ciphertextLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the wrapped key.
	Variation	–
	authenticatorPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	Contains the pointer to the data of the authenticator.
	Variation	–
	authenticatorLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the authenticator.
	Variation	–
	verifyPtr	
	Type	Csm_VerifyResultPtr
	Direction	IN
	Comment	Contains the verification result.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY CRYPTO_E_KEY_WRITE_FAIL CRYPTO_E_KEY_NOT_VALID CRYPTO_E_KEY_SIZE_MISMATCH CRYPTO_E_JOB_CANCELED	

⌋

8.6.4 Client-Server-Interfaces (Context Service)

[SWS_Csm_91106] Definition of ClientServerInterface CsmContextService_{Job} ⌈

Name	CsmContextService_{Job}		
Comment	Interface to context data operation		
IsService	true		
Variation	{ecuc(Csm/CsmJobs/CsmJob. CsmJobServiceInterfaceContextUsePort)} == CRYPTO_USE_PORT Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)}		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	CRYPTO_E_BUSY	Request failed, service is still busy.

Operation	RestoreContextJob	
Comment	Restore the job context.	
Relates to	Csm_RestoreContextJob	
Variation	–	
Parameters	contextBufferPtr	
	Type	ConstVoidPtr
	Direction	IN
	Comment	Provides the buffer for the context as [DATA_REFERENCE].
	Variation	–
	contextBufferLength	
	Type	uint32
	Direction	IN
	Comment	Contains the length in bytes of the context buffer.
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY	

Operation	SaveContextJob	
Comment	Save the job context to the provided buffer.	
Relates to	Csm_SaveContextJob	
Variation	–	
Parameters	contextBufferPtr	
	Type	VoidPtr
	Direction	IN
	Comment	Provides the buffer for the context as [DATA_REFERENCE].
	Variation	–
	contextBufferLengthPtr	
	Type	Csm_LengthPtr
	Direction	IN
	Comment	Contains the length in bytes of the context buffer as [DATA_REFERENCE].
	Variation	–
Possible Errors	E_OK E_NOT_OK CRYPTO_E_BUSY	

⌋

8.6.5 Client-Server-Interface Callbacks

[SWS_Csm_00928] Definition of ClientServerInterface CallbackNotification ⌈

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

Operation	CallbackNotification	
Comment	Notifies the application with a return value that the job has finished.	
Relates to	–	
Variation	–	
Parameters	result	
	Type	Crypto_ResultType
	Direction	IN
	Comment	Return value that shall be returned to the application
	Variation	–
Possible Errors	–	

⌋

8.6.6 Implementation Data Types

[SWS_Csm_01031] Definition of ImplementationDataType Crypto_ServiceInfo Type ⌈

Name	Crypto_ServiceInfoType		
Kind	Type		
Derived from	uint8		
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_KEYEXCHANGE-CALCPUBVAL	0x0F	KeyExchangeCalcPubVal Service
	CRYPTO_KEYEXCHANGE-CALCSECRET	0x10	KeyExchangeCalcSecret Service
	CRYPTO_KEYSETVALID	0x13	KeySetValid Service
	CRYPTO_KEYSETINVALID	0x14	KeySetInvalid Service
	CRYPTO_CUSTOM_ SERVICE	0x15	Custom service job
	CRYPTO_KEYWRAP	0x16	KeyWrap Service
	CRYPTO_KEYUNWRAP	0x17	KeyUnwrap Service
Description	Enumeration of the kind of the service.		
Variation	–		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_01029] Definition of ImplementationDataType Crypto_Operation ModeType ⌈

Name	Crypto_OperationModeType		
Kind	Type		
Derived from	uint8		
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".
	CRYPTO_ OPERATIONMODE_SAVE_ CONTEXT	0x08	Operation mode is “Save workspace context”. Context data shall be provided by the crypto driver to the application.
	CRYPTO_ OPERATIONMODE_ RESTORE_CONTEXT	0x10	Operation mode is “Restore workspace context”. Application provides the context data that was previously stored and the crypto driver shall restore the internal workspace.
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		

⌋

[SWS_Csm_01049] Definition of ImplementationDataType Crypto_ProcessingType ⌈

Name	Crypto_ProcessingType		
Kind	Type		
Derived from	uint8		
Range	CRYPTO_PROCESSING_ASYNC	0x00	Asynchronous job processing
	CRYPTO_PROCESSING_SYNC	0x01	Synchronous job processing
Description	Enumeration of the processing type.		
Variation	–		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_01024] Definition of ImplementationDataType Crypto_VerifyResultType ⌈

Name	Crypto_VerifyResultType		
Kind	Type		
Derived from	uint8		
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	Rte_Csm_Type.h		

⌋

[SWS_Csm_91044] Definition of ImplementationDataType Crypto_ResultType ⌈

Name	Crypto_ResultType		
Kind	Type		
Derived from	uint8		
Range	E_OK	0x00	The service request is successful.
	E_NOT_OK	0x01	The service request failed.
	CRYPTO_E_BUSY	0x02	The service request failed because the service is still busy
	CRYPTO_E_ENTROPY_EXHAUSTED	0x04	The service request failed because the entropy of the random number generator is exhausted
	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_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.
	CRYPTO_E_CUSTOM_ERROR	0x0E	Custom processing failed.
Description	Return for Std_ReturnType for Cryptostack.		
Variation	–		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_91121] Definition of ImplementationDataType Crypto_DataInputType

⌈

Name	Crypto_DataInputType	
Kind	Structure	
Elements	dataPtr	
	Type	const uint8*
	Comment	Pointer to a memory location that stores input data.
	dataLength	
	Type	uint32
	Comment	Contains the relevant length of the memory location. Whether the length is interpreted as BITS or BYTES depends on the Csm job's service.
	keyId	
	Type	uint32
	Comment	Identifier of the Crypto key, mapped from the CryIf key (== job->config->jobInputRedirectionsPtr[Idx].keyId), from which an element shall be used as alternative input or CRYPTO_NO_ID if no key is specified.
Description	Structure which contains information regarding an input of a Csm job in general.	
Variation	–	
Available via	Rte_Csm_Type.h	

⌋

[SWS_Csm_91122] Definition of ImplementationDataType Crypto_DataOutputType ⌈

Name	Crypto_DataOutputType	
Kind	Structure	
Elements	dataPtr	
	Type	uint8*
	Comment	Pointer to a memory location that shall store output data.
	dataLengthPtr	
	Type	uint32*
	Comment	Pointer to a memory location that shall store 1. initially the length in BYTES of the memory location that shall store the output data 2. later the actually used length in BYTES.
	keyId	
	Type	uint32
	Comment	Identifier of the Crypto key, mapped from the CryIf key (== job->config->jobOutputRedirectionsPtr[Idx].keyId), from which an element shall be used as alternative output or CRYPTO_NO_ID if no key is specified.
Description	Structure which contains information regarding an output of a Csm job in general.	
Variation	–	
Available via	Rte_Csm_Type.h	

⌋

[SWS_Csm_91123] Definition of ImplementationDataType Crypto_DataRedirectionType ⌈

Name	Crypto_DataRedirectionType	
Kind	Structure	
Elements	keyId	
	Type	uint32
	Comment	Identifier of the CryIf key, one of whose key elements shall be used as a replacement for an input or output, which would otherwise be specified by the application.
	keyElementId	
	Type	uint32
	Comment	Identifier of the key element which shall be used as a replacement for an input or output, which would otherwise be specified by the application.
Description	Structure which contains information regarding an alternative location (key and key element) for an input or output to a Csm job.	
Variation	–	
Available via	Rte_Csm_Type.h	

⌋

[SWS_Csm_91124] Definition of ImplementationDataType Crypto_JobConfig Type ⌈

Name	Crypto_JobConfigType	
Kind	Structure	
Elements	jobInputRedirectionsPtr	
	Type	const Crypto_DataRedirectionType* const
	Comment	Pointer to a constant array of size CRYPTO_MAX_INPUTS and type Crypto_DataRedirectionType that describe redirections for the individual Csm job inputs or NULL_PTR if no input redirection is used at all. A single inactive redirection within the array of redirections is identified by all members of Crypto_DataRedirectionType set to CRYPTO_NO_ID.
	jobOutputRedirectionsPtr	
	Type	const Crypto_DataRedirectionType* const
	Comment	Pointer to a constant array of size CRYPTO_MAX_OUTPUTS and type Crypto_DataRedirectionType that describe redirections for the individual Csm job outputs or NULL_PTR if no output redirection is used at all. A single inactive redirection within the array of redirections is identified by all members of Crypto_DataRedirectionType set to CRYPTO_NO_ID.
	jobId	
	Type	uint32
	Comment	Identifier for the Csm job.
	jobPriority	
	Type	uint32
	Comment	Specifies the importance of the Csm job (the higher, the more important). (=> Csm/CsmJobs/CsmJob/CsmJobPriority)
	jobPrimitiveId	
	Type	uint32
	Comment	Identifier of the CryIf primitive assigned to the Csm primitive assigned to the Csm job. (=> Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef -> Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveRef -> CryIf/CryIfPrimitives/CryIfPrimitive/CryIfPrimitiveId)
	jobKeyId	
	Type	uint32
	Comment	Identifier of the CryIf key assigned to the Csm key assigned to the job or CRYPTO_NO_ID if no key is assigned. (=> Csm/CsmJobs/CsmJob/CsmJobKeyRef -> Csm/CsmKeys/CsmKey/CsmKeyRef -> CryIf/CryIfKeys/CryIfKey/CryIfKeyId)
	jobCallbackId	
	Type	uint32
	Comment	Internal identifier of the Csm callback function, to be called, if the job is finished or CRYPTO_NO_ID if no callback is assigned.
	jobProcessingType	
	Type	Crypto_ProcessingType
	Comment	Determines the synchronous or asynchronous behavior. (=> Csm/CsmJobs/CsmJob/CsmProcessingMode)
	jobService	

▽

△

	Type	Crypto_ServiceInfoType
	Comment	Contains the enum of the used service, e.g. CRYPTO_ENCRYPT. (=> `Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef` -> `Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveRef` -> `CryIf/CryIfPrimitives/CryIfPrimitive/CryIfPrimitiveRef` -> `Crypto/CryptoPrimitives/CryptoPrimitive/CryptoPrimitiveTypeRef` -> `Crypto/CryptoPrimitiveTypes/CryptoPrimitiveType/CryptoPrimitiveTypeService`)
	jobInputsCount	
	Type	uint8
	Comment	Size of the array jobInputsPtr points to. The size depends on the number of specified inputs, respectively generally necessary inputs of a service. E.g. ENCRYPT = 1, MACVERIFY = 2. See [SWS_Crypto_00071] and [SWS_Crypto_00073].
	jobOutputsCount	
	Type	uint8
	Comment	Size of the array jobOutputsPtr points to. The size depends on the number of specified outputs, respectively generally necessary outputs of a service. E.g. ENCRYPT = 1, AEADENCRYPT = 2. See [SWS_Crypto_00071] and [SWS_Crypto_00073].
Description	Structure which contains the run-time invariant data regarding a specific Csm job as it is directly or indirectly derived from the configuration.	
Variation	–	
Available via	Rte_Csm_Type.h	

⌋

[SWS_Csm_91125] Definition of ImplementationDataType Crypto_JobActionType ⌈

Name	Crypto_JobActionType	
Kind	Structure	
Elements	jobInputsPtr	
	Type	Crypto_DataInputType*
	Comment	Pointer to an array of size `jobInputsCount` and type `Crypto_DataInputType` that describes input buffers and/or redirection keyIds for the individual job inputs or NULL_PTR if no inputs are used at all.
	jobOutputsPtr	
	Type	Crypto_DataOutputType*
	Comment	Pointer to an array of size `jobOutputsCount` and type `Crypto_DataOutputType` that describe output buffers and/or redirection keyIds for the individual job outputs or NULL_PTR if no outputs are used at all.
	jobOutputVerifyPtr	
	Type	Crypto_VerifyResultType*
	Comment	Pointer to a memory location holding a variable of type `Crypto_VerifyResultType`.
	jobPrimitiveId	
	Type	uint32

▽

△

	Comment	Identifier of the Crypto primitive Id mapped from the CryIf primitive Id (`Crypto_JobConfigType->jobPrimitiveId`) assigned to the Csm primitive assigned to the Csm job.
	jobKeyId	
	Type	uint32
	Comment	Identifier of the Crypto key Id mapped from the CryIf key Id (`Crypto_JobConfigType->jobKeyId`) assigned to the Csm primitive assigned to the Csm job or `CRYPTO_NO_ID` if no key is assigned.
	jobTargetKeyId	
	Type	uint32
	Comment	Identifier of the mapped CryIf key and then mapped Crypto key that was passed as parameter 1. `targetKeyId` to API `Csm_JobKeyDerive` or 2. `targetKeyId` to API `Csm_CustomService` or `CRYPTO_NO_ID` if no key is assigned.
	jobOperationMode	
	Type	Crypto_OperationModeType
	Comment	The operation mode to be used with the job.
Description	Structure which contains the run-time variable data regarding a specific Csm job.	
Variation	–	
Available via	Rte_Csm_Type.h	

⌋

[SWS_Csm_91126] Definition of ImplementationDataType Crypto_JobType ⌈

Name	Crypto_JobType	
Kind	Structure	
Elements	config	
	Type	const Crypto_JobConfigType*
	Comment	Pointer to a structure containing run-time invariant data regarding a Csm job.
	action	
	Type	Crypto_JobActionType*
	Comment	Pointer to a structure containing run-time variable data regarding a Csm job.
Description	Structure which contains information, that describes a Csm job.	
Variation	–	
Available via	Rte_Csm_Type.h	

⌋

[SWS_Csm_00828] Definition of ImplementationDataType Csm_KeyDataType_{Crypto} ⌈

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 any key element of the considered key		
Variation	Crypto = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_91119] Definition of ImplementationDataType Csm_{Service}DataType_{Primitive} ⌈

Name	Csm_{Service}DataType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveInput0MaxLength)} Elements Elements		
Description	Array long enough to store the 1. ciphertext (Decrypt) 2. plaintext (Encrypt) 3. data (Hash, MacGenerate, MacVerify, SignatureGenerate, SignatureVerify) that shall be processed.		
Variation	Service={ecuc/Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef->Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveRef->CryIf/CryIfPrimitives/CryIfPrimitive/CryIfPrimitiveRef->Crypto/CryptoPrimitives/CryptoPrimitive/CryptoPrimitiveTypeRef->Crypto/CryptoPrimitiveTypes/CryptoPrimitiveType/CryptoPrimitiveTypeService} in [CRYPTO_DECRYPT, CRYPTO_ENCRYPT, CRYPTO_HASH, CRYPTO_MACGENERATE, CRYPTO_MACVERIFY, CRYPTO_SIGNATUREGENERATE, CRYPTO_SIGNATUREVERIFY] Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_01925] Definition of ImplementationDataType Csm_AEADEncryptPlaintextType_{Primitive} ⌈

Name	Csm_AEADEncryptPlaintextType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveInput0MaxLength)} Elements		
Description	Array long enough to store the plaintext to be encrypted with AEAD.		
Variation	Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_01927] Definition of ImplementationDataType Csm_AEADEncryptCiphertextType_{Primitive} ⌈

Name	Csm_AEADEncryptCiphertextType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveOutput0MaxLength)} Elements		
Description	Array long enough to store the data of the cipher.		
Variation	Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_01926] Definition of ImplementationDataType Csm_AEADEncryptTagType_{Primitive} ⌈

Name	Csm_AEADEncryptTagType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveOutput1MaxLength)} Elements		
Description	Array long enough to store the data of the Tag.		
Variation	Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_00922] Definition of ImplementationDataType Csm_AEADDecryptCiphertextType_{Primitive} ⌈

Name	Csm_AEADDecryptCiphertextType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveInput0MaxLength)} Elements		
Description	Array long enough to store the ciphertext to be decrypted with AEAD.		
Variation	Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_91127] Definition of ImplementationDataType Csm_{Service}AssociatedDataType_{Primitive} ⌈

Name	Csm_{Service}AssociatedDataType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveInput1MaxLength)} Elements		
Description	Array long enough to store the 1. associated data (AEADDecrypt, AEADEncrypt) to be used.		

▽

△

Variation	Service={ecuc/Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef->Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveRef->CryIf/CryIfPrimitives/CryIfPrimitive/CryIfPrimitiveRef->Crypto/CryptoPrimitives/CryptoPrimitive/CryptoPrimitiveTypeRef->Crypto/CryptoPrimitiveTypes/CryptoPrimitiveType/CryptoPrimitiveTypeService} in [CRYPTO_AEADDECRYPT, CRYPTO_AEADENCRYPT] Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}
Available via	Rte_Csm_Type.h

⌋

[SWS_Csm_01074] Definition of ImplementationDataType Csm_AEADDecryptTagType_{Primitive} ⌈

Name	Csm_AEADDecryptTagType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveInput2MaxLength)} Elements		
Description	Array long enough to store the data of the Tag.		
Variation	Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_01075] Definition of ImplementationDataType Csm_AEADDecryptPlaintextType_{Primitive} ⌈

Name	Csm_AEADDecryptPlaintextType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveOutput0MaxLength)} Elements		
Description	Array long enough to store the data of the plaintext.		
Variation	Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_91128] Definition of ImplementationDataType Csm_{Service}CompareType_{Primitive} ⌈

Name	Csm_{Service}CompareType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveInput1MaxLength)} Elements		
Description	Array long enough to store the 1. result (MacVerify, SignatureVerify) to be verified.		
Variation	Service={ecuc/Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef->Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveRef->CryIf/CryIfPrimitives/CryIfPrimitive/CryIfPrimitiveRef->Crypto/CryptoPrimitives/CryptoPrimitive/CryptoPrimitiveTypeRef->Crypto/CryptoPrimitiveTypes/CryptoPrimitiveType/CryptoPrimitiveTypeService} in [CRYPTO_MACVERIFY, CRYPTO_SIGNATUREVERIFY] Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_91120] Definition of ImplementationDataType Csm_{Service}ResultType_{Primitive} ⌈

Name	Csm_{Service}ResultType_{Primitive}		
Kind	Array	***Element type***	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveOutput0MaxLength)} Elements Elements		
Description	Array long enough to store the 1. result (Decrypt, Encrypt, Hash, MacGenerate, RandomGenerate, SignatureGenerate) that shall be calculated.		
Variation	Service={ecuc/Csm/CsmJobs/CsmJob/CsmJobPrimitiveRef->Csm/CsmPrimitives/CsmPrimitive/CsmPrimitiveRef->CryIf/CryIfPrimitives/CryIfPrimitive/CryIfPrimitiveRef->Crypto/CryptoPrimitives/CryptoPrimitive/CryptoPrimitiveTypeRef->Crypto/CryptoPrimitiveTypes/CryptoPrimitiveType/CryptoPrimitiveTypeService} in [CRYPTO_DECRYPT, CRYPTO_ENCRYPT, CRYPTO_HASH, CRYPTO_MACGENERATE, CRYPTO_RANDOMGENERATE, CRYPTO_SIGNATUREGENERATE] Primitive={ecuc/Csm/CsmPrimitives/CsmPrimitive.SHORT-NAME}		
Available via	Rte_Csm_Type.h		

⌋

[SWS_Csm_91045] Definition of ImplementationDataType Csm_LengthPtr ⌈

Name	Csm_LengthPtr
Kind	Pointer
Type	uint32*
Description	ImplementationDataType of category DATA_REFERENCE with the pointer target uint32 to be able to return asynchronously the length.
Variation	–
Available via	Rte_Csm_Type.h

⌋

[SWS_Csm_91046] Definition of ImplementationDataType Csm_VerifyResultPtr ⌈

Name	Csm_VerifyResultPtr
Kind	Pointer
Type	Crypto_VerifyResultType*
Description	ImplementationDataType of category DATA_REFERENCE with the pointer target Crypto_VerifyResultType to be able to return asynchronously the verify result.
Variation	–
Available via	Rte_Csm_Type.h

⌋

[SWS_Csm_91102] Definition of ImplementationDataType Crypto_KeyStatusType ⌈

Name	Crypto_KeyStatusType
Kind	Type
Derived from	uint8

▽

△

Range	CRYPTO_KEYSTATUS_INVALID	0x00	The status of the key is invalid (for example after Csm_KeyElement Set the Csm_KeySetValid was not called).
	CRYPTO_KEYSTATUS_VALID	0x01	The status of the key is valid (for example the status was successfully set by the Csm_Key SetValid).
	CRYPTO_KEYSTATUS_UPDATE_IN_PROGRESS	0x02	Indicates that the NV RAM Block that is assigned to this key are currently being updated. The update operation is in progress and has not yet been completed by NVM.
Description	Enumeration for key status.		
Variation	–		
Available via	Rte_Csm_Type.h		

⌋

8.6.7 Ports

[SWS_Csm_01042] Definition of Port CsmKey_{Key} provided by module Csm ⌈

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)}		

⌋

[SWS_Csm_91023] Definition of Port CsmJob_{Job} provided by module Csm ⌈

Name	CsmJob_{Job}		
Kind	ProvidedPort	***Interface***	CsmAEADDecrypt_{Primitive}, CsmAEADEncrypt_{Primitive}, CsmDecrypt_{Primitive}, CsmEncrypt_{Primitive}, CsmHash_{Primitive}, CsmMacGenerate_{Primitive}, CsmMacVerify_{Primitive}, CsmRandomGenerate_{Primitive}, CsmSignatureGenerate_{Primitive}, CsmSignatureVerify_{Primitive}
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)}

⌋

[SWS_Csm_91062] Definition of Port CsmJob_{Job} provided by module Csm ⌈

Name	CsmJob_{Job}		
Kind	ProvidedPort	***Interface***	CsmAEADDecrypt, CsmAEADEncrypt, CsmCustom, CsmDecrypt, CsmEncrypt, CsmHash, CsmMac Generate, CsmMacVerify, CsmSignatureGenerate, Csm SignatureVerify
Description	Port related to a specific cryptographic job to execute the assigned cryptographic calculations synchronously or 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) Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)}		

⌋

[SWS_Csm_91112] Definition of Port CsmJob_{Job} provided by module Csm ⌈

Name	CsmJob_{Job}		
Kind	ProvidedPort	***Interface***	CsmJobKeyDerive, CsmJobKeyExchangeCalcPubVal, CsmJobKeyExchangeCalcSecret, CsmJobKeyGenerate, CsmJobKeySetInvalid, CsmJobKeySetValid, CsmJobKey Unwrap, CsmJobKeyWrap, CsmJobRandomSeed, Csm RandomGenerate
Description	Port related to a specific cryptographic job to execute the assigned cryptographic calculations synchronously or asynchronously (no Operation Mode).		
Port Defined Argument Value(s)	***Type***	uint32	
	Value	{ecuc(Csm/CsmJobs/CsmJob.CsmJobId)}	
Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmJobInterfaceUsePort)} == CRYPTO_USE_PORT_ OPTIMIZED) Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)}		

⌋

[SWS_Csm_00934] Definition of Port CallbackNotification_{Job} required by module Csm ⌈

Name	CallbackNotification_{Job}		
Kind	RequiredPort	***Interface***	CallbackNotification
Description	Port for the callback notification.		

▽

△

Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmProcessingMode)}==CRYPTO_PROCESSING_ASYN)&&(CsmJob/CsmJobInterfaceUsePort!=CRYPTO_USE_FNC) Job = {ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)}

⌋

[SWS_Csm_91105] Definition of Port CsmContext_{Job} provided by module Csm ⌈

Name	CsmContext_{Job}		
Kind	ProvidedPort	***Interface***	CsmContextService_{Job}
Description	Port related to context data operation.		
Port Defined Argument Value(s)	***Type***	uint32	
	Value	{ecuc(Csm/CsmJobs/CsmJob.CsmJobId)}	
Variation	{ecuc(Csm/CsmJobs/CsmJob. CsmJobServiceInterfaceContextUsePort)} == CRYPTO_USE_PORT Job={ecuc(Csm/CsmJobs/CsmJob.SHORT-NAME)}		

⌋

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 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).

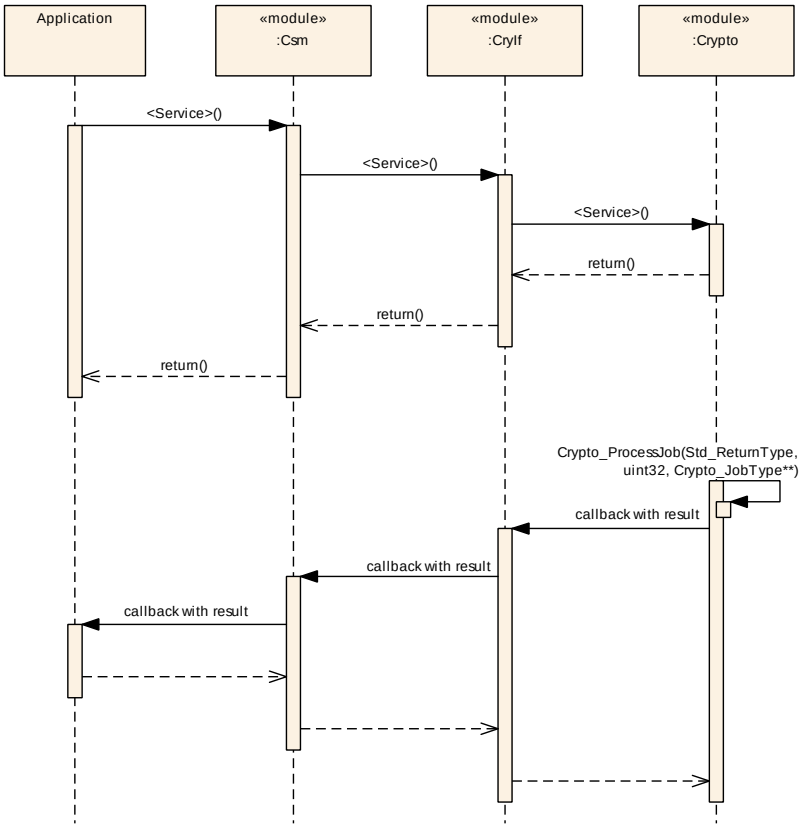


Figure 9.1: Sequence Diagram for Asynchronous Call with Callback

9.2 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.

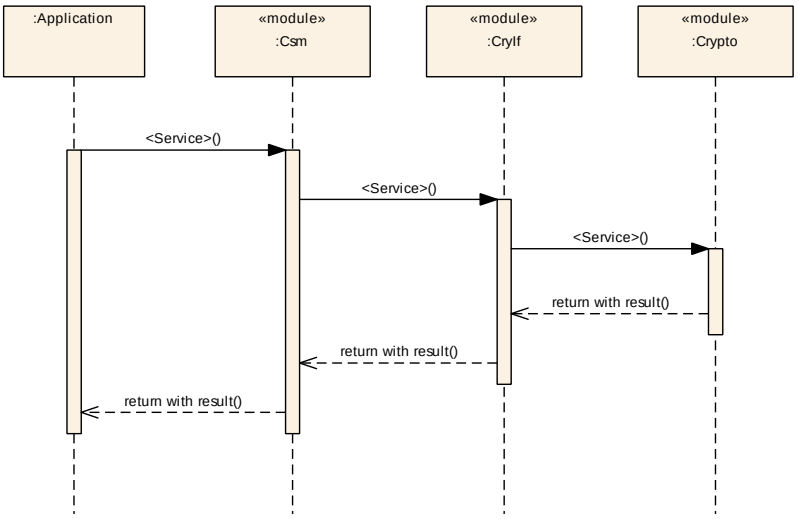


Figure 9.2: Sequence Diagram for Synchronous Call

10 Configuration specification

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. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

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 [5] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

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

Note: The Ids in the configuration containers shall be consecutive, gapless and shall start from zero.

10.2.1 Csm

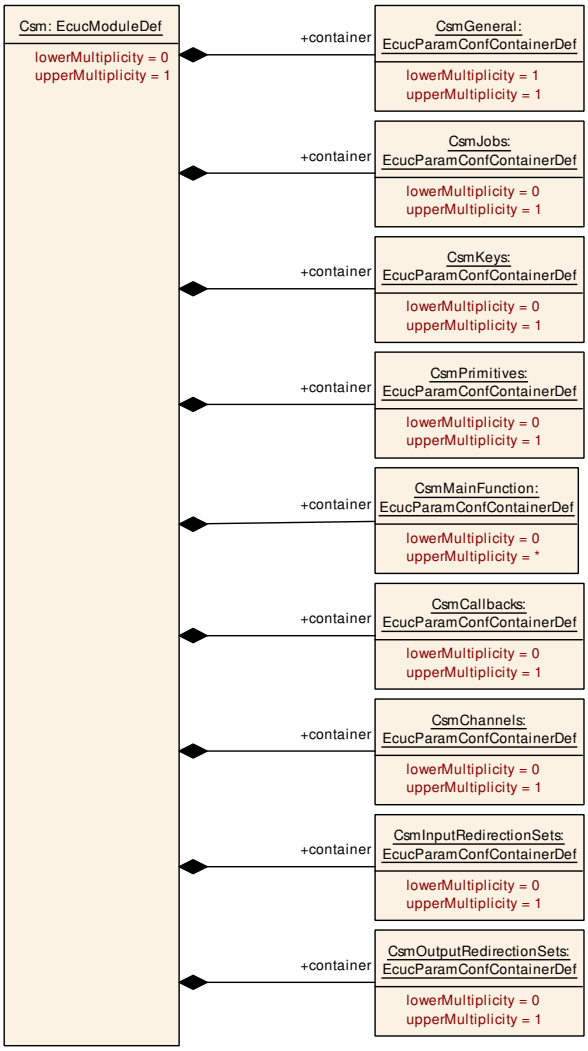


Figure 10.1: Crypto Service Manager Layout

[ECUC_Csm_00818] Definition of EcucModuleDef Csm ⌈

Module Name	Csm
Description	Configuration of the Csm (CryptoServiceManager) module.
Post-Build Selectable Support	false
Post-Build Loadable Support	false

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmCallbacks	0..1	Container for callback function configurations
CsmChannels	0..1	Container for CSM channel configurations
CsmGeneral	1	Container for common configuration options.
CsmInputRedirectionSets	0..1	Container for configuration of sets of CSM input redirections.
CsmJobs	0..1	Container for configuration of CSM jobs.

▽

△

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmKeys	0..1	Container for CSM key configurations.
CsmMainFunction	0..*	Each element of this container defines one instance of Csm_MainFunction. For each partition, where the Csm module shall be instantiated, at least one MainFunction instance needs to be configured.
CsmOutputRedirectionSets	0..1	Container for configuration of sets of CSM output redirections.
CsmPrimitives	0..1	Container for configuration of CsmPrimitives

⌋

10.2.2 CsmGeneral

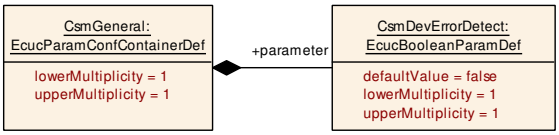


Figure 10.2: Crypto Service Manager General Layout

[ECUC_Csm_00002] Definition of EcucParamConfContainerDef CsmGeneral ⌈

Container Name	CsmGeneral
Parent Container	Csm
Upstream mapping defined	–
Description	Container for common configuration options.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CsmDevErrorDetect	1	[ECUC_Csm_00001]

No Included Containers

⌋

[ECUC_Csm_00001] Definition of EcucBooleanParamDef CsmDevErrorDetect ⌈

Parameter Name	CsmDevErrorDetect
Parent Container	CsmGeneral
Upstream mapping defined	–

▽

△

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
Pre-Compile only	true

⌋

10.2.3 CsmChannels

[ECUC_Csm_00376] Definition of EcucParamConfContainerDef CsmChannels ⌈

Container Name	CsmChannels
Parent Container	Csm
Upstream mapping defined	–
Description	Container for CSM channel configurations
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmChannel	1..*	Container for configuration of a CSM channel.

⌋

10.2.4 CsmChannel

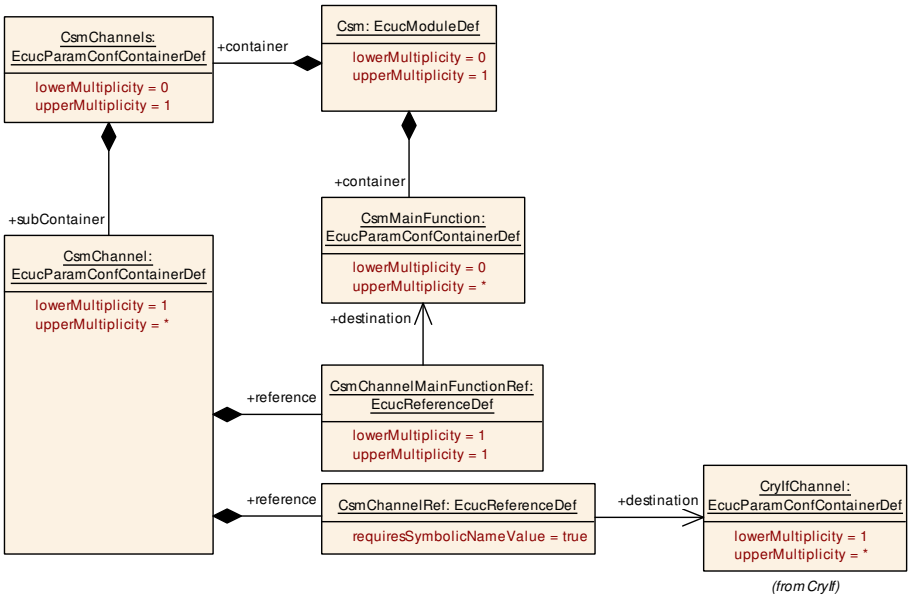


Figure 10.3: Crypto Service Manager Channel Layout

[ECUC_Csm_00379] Definition of EcucParamConfContainerDef CsmChannel ⌈

Container Name	CsmChannel
Parent Container	CsmChannels
Upstream mapping defined	–
Description	Container for configuration of a CSM channel.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CsmChannelMainFunctionRef	1	[ECUC_Csm_00380]
CsmChannelRef	1	[ECUC_Csm_00033]

No Included Containers

⌋

[ECUC_Csm_00380] Definition of EcucReferenceDef CsmChannelMainFunction Ref ⌈

Parameter Name	CsmChannelMainFunctionRef
Parent Container	CsmChannel
Upstream mapping defined	–
Description	Reference to CsmMainFunction, where the according CsmChannel is assigned to.
Multiplicity	1

▽

△

Type	Reference to CsmMainFunction
Pre-Compile only	true

⌋

[ECUC_Csm_00033] Definition of EcucReferenceDef CsmChannelRef ⌈

Parameter Name	CsmChannelRef
Parent Container	CsmChannel
Upstream mapping defined	–
Description	Refers to the underlying Crypto Interface channel.
Multiplicity	1
Type	Symbolic name reference to CryIfChannel
Pre-Compile only	true

⌋

10.2.5 CsmMainFunction

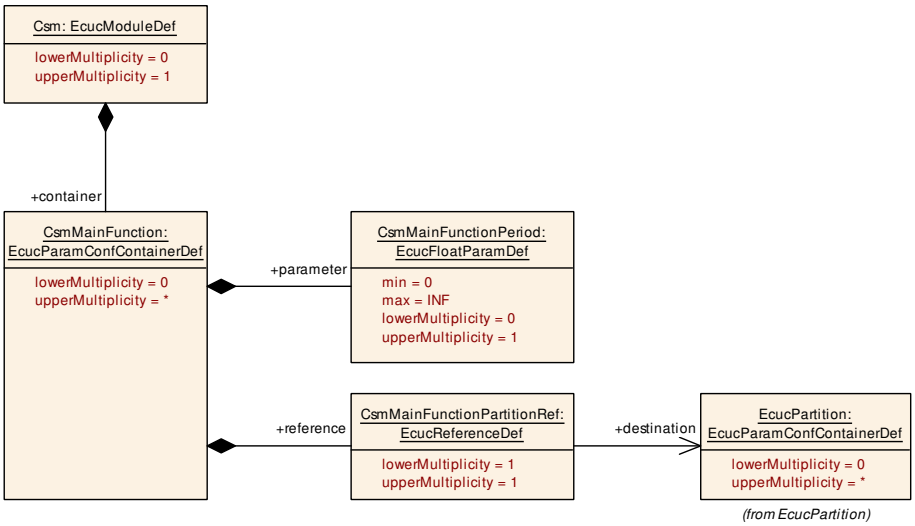


Figure 10.4: Crypto Service Manager MainFunction Layout

[ECUC_Csm_00279] Definition of EcucParamConfContainerDef CsmMainFunction ⌈

Container Name	CsmMainFunction
Parent Container	Csm
Upstream mapping defined	–
Description	Each element of this container defines one instance of Csm_MainFunction. For each partition, where the Csm module shall be instantiated, at least one MainFunction instance needs to be configured.
Multiplicity	0..*

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CsmMainFunctionPeriod	0..1	[ECUC_Csm_00113]
CsmMainFunctionPartitionRef	1	[ECUC_Csm_00280]

No Included Containers

⌋

[ECUC_Csm_00113] Definition of EcucFloatParamDef CsmMainFunctionPeriod ⌈

Parameter Name	CsmMainFunctionPeriod	
Parent Container	CsmMainFunction	
Upstream mapping defined	–	
Description	Specifies the period of main function Csm_MainFunction in seconds.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00280] Definition of EcucReferenceDef CsmMainFunctionPartitionRef ⌈

Parameter Name	CsmMainFunctionPartitionRef
Parent Container	CsmMainFunction
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according CsmMainFunction instance is assigned to.
Multiplicity	1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

10.2.6 CsmJobs

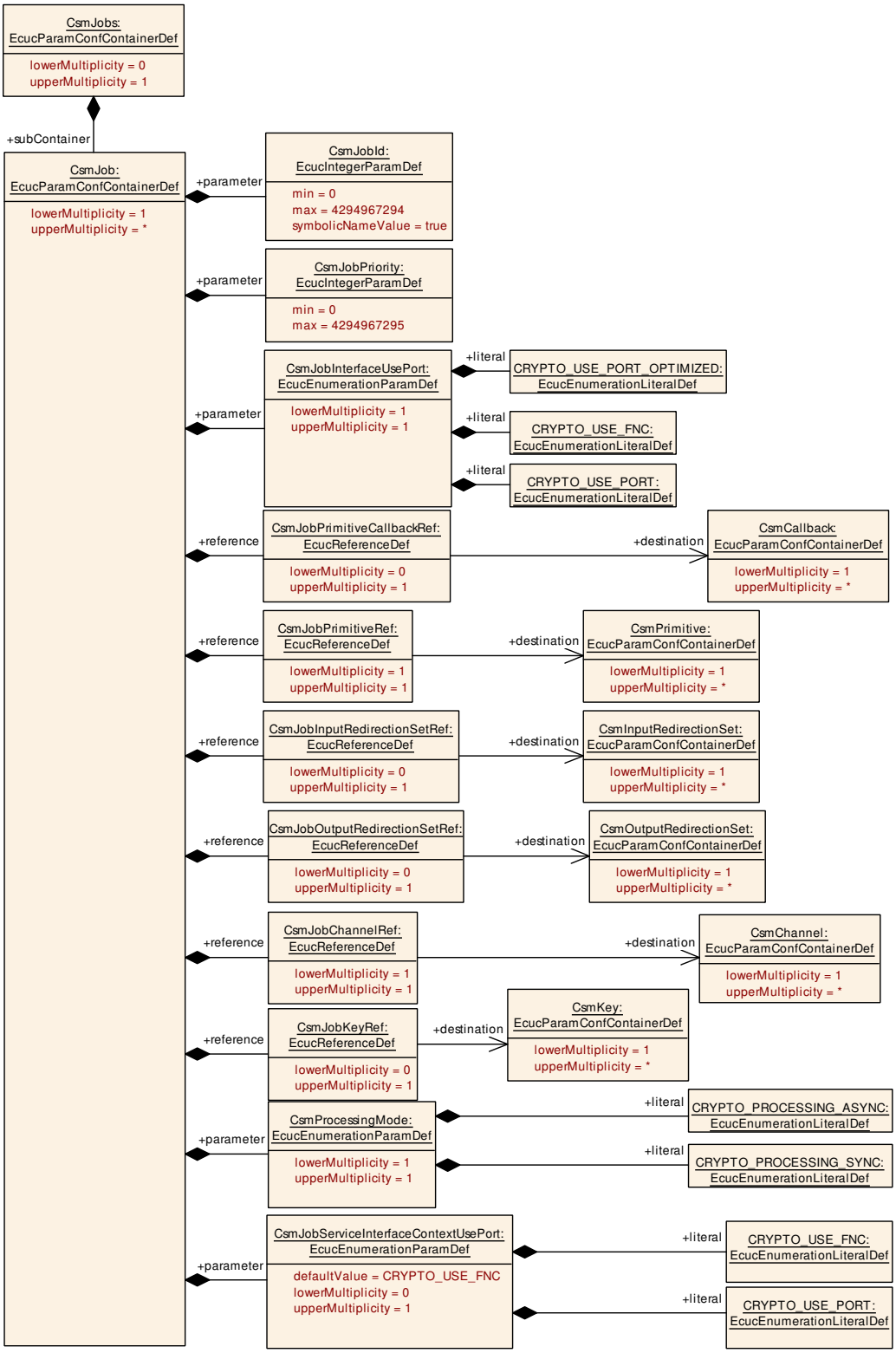


Figure 10.5: CsmJobs Layout

[ECUC_Csm_00112] Definition of EcucParamConfContainerDef CsmJobs ⌈

Container Name	CsmJobs
Parent Container	Csm
Upstream mapping defined	–
Description	Container for configuration of CSM jobs.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

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

⌋

10.2.7 CsmJob

[ECUC_Csm_00118] Definition of EcucParamConfContainerDef CsmJob ⌈

Container Name	CsmJob
Parent Container	CsmJobs
Upstream mapping defined	–
Description	Container for configuration of CSM job. The container name serves as a symbolic name for the identifier of a job configuration.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CsmJobId	1	[ECUC_Csm_00119]
CsmJobInterfaceUsePort	1	[ECUC_Csm_00275]
CsmJobPriority	1	[ECUC_Csm_00120]
CsmJobServiceInterfaceContextUsePort	0..1	[ECUC_Csm_00327]
CsmProcessingMode	1	[ECUC_Csm_00276]
CsmJobChannelRef	1	[ECUC_Csm_00399]
CsmJobInputRedirectionSetRef	0..1	[ECUC_Csm_00397]
CsmJobKeyRef	0..1	[ECUC_Csm_00126]
CsmJobOutputRedirectionSetRef	0..1	[ECUC_Csm_00398]
CsmJobPrimitiveCallbackRef	0..1	[ECUC_Csm_00123]
CsmJobPrimitiveRef	1	[ECUC_Csm_00122]

No Included Containers

⌋

[ECUC_Csm_00119] Definition of EcucIntegerParamDef CsmJobId ⌈

Parameter Name	CsmJobId	
Parent Container	CsmJob	
Upstream mapping defined	–	
Description	Identifier of the CSM job. Dependency: The range is limited to 0xFFFFFFFE, to have the value 0xFFFFFFFF (CRYPTO_NO_ID) reserved for stack internal use. This is common for Crypto stack "Id" parameters.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 4294967294	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00275] Definition of EcucEnumerationParamDef CsmJobInterfaceUsePort ⌈

Parameter Name	CsmJobInterfaceUsePort	
Parent Container	CsmJob	
Upstream mapping defined	–	
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.
Pre-Compile only	true	

⌋

[ECUC_Csm_00120] Definition of EcucIntegerParamDef CsmJobPriority ⌈

Parameter Name	CsmJobPriority	
Parent Container	CsmJob	
Upstream mapping defined	–	
Description	Priority of the job. The higher the value, the higher the job's priority.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00327] Definition of EcucEnumerationParamDef CsmJobServiceInterfaceContextUsePort ⌈

Parameter Name	CsmJobServiceInterfaceContextUsePort	
Parent Container	CsmJob	
Upstream mapping defined	–	
Description	Does the job need RTE interfaces for context operations	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	CRYPTO_USE_FNC	Port is not used.
	CRYPTO_USE_PORT	Port is used for this operation.
Default value	CRYPTO_USE_FNC	
Pre-Compile only	true	

⌋

[ECUC_Csm_00276] Definition of EcucEnumerationParamDef CsmProcessingMode ⌈

Parameter Name	CsmProcessingMode	
Parent Container	CsmJob	
Upstream mapping defined	–	
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	Asynchronous processing
	CRYPTO_PROCESSING_SYNC	Synchronous processing
Pre-Compile only	true	

⌋

[ECUC_Csm_00399] Definition of EcucReferenceDef CsmJobChannelRef ⌈

Parameter Name	CsmJobChannelRef
Parent Container	CsmJob
Upstream mapping defined	–
Description	This parameter refers to the CsmChannel.
Multiplicity	1
Type	Reference to CsmChannel
Pre-Compile only	true

⌋

[ECUC_Csm_00397] Definition of EcucReferenceDef CsmJobInputRedirectionSetRef ⌈

Parameter Name	CsmJobInputRedirectionSetRef
Parent Container	CsmJob
Upstream mapping defined	–
Description	This parameter refers to the Csm input redirection set that shall be used for this Csm job.
Multiplicity	0..1
Type	Reference to CsmInputRedirectionSet
Pre-Compile only	true

⌋

[ECUC_Csm_00126] Definition of EcucReferenceDef CsmJobKeyRef ⌈

Parameter Name	CsmJobKeyRef
Parent Container	CsmJob
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This parameter refers to the key which shall be used for the CsmPrimitive. It's possible to use a CsmKey for different jobs Dependency: If the CsmPrimitive does not rely on a key (e.g. Hash or Compression/ Decompression via Encrypt/Decrypt), then this does not need to be configured.
Multiplicity	0..1
Type	Reference to CsmKey
Pre-Compile only	true

⌋

[ECUC_Csm_00398] Definition of EcucReferenceDef CsmJobOutputRedirectionSetRef ⌈

Parameter Name	CsmJobOutputRedirectionSetRef
Parent Container	CsmJob
Upstream mapping defined	–
Description	This parameter refers to the CSM output redirection set that shall be used for this Csm job.
Multiplicity	0..1
Type	Reference to CsmOutputRedirectionSet
Pre-Compile only	true

⌋

[ECUC_Csm_00123] Definition of EcucReferenceDef CsmJobPrimitiveCallbackRef ⌈

Parameter Name	CsmJobPrimitiveCallbackRef
Parent Container	CsmJob
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_Csm_00122] Definition of EcucReferenceDef CsmJobPrimitiveRef ⌈

Parameter Name	CsmJobPrimitiveRef
Parent Container	CsmJob
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This parameter refers to the used CsmPrimitive. Different jobs may refer to one Csm Primitive. The referred CsmPrimitive provides detailed information on the actual cryptographic routine.
Multiplicity	1
Type	Reference to CsmPrimitive
Pre-Compile only	true

⌋

10.2.8 CsmKeys

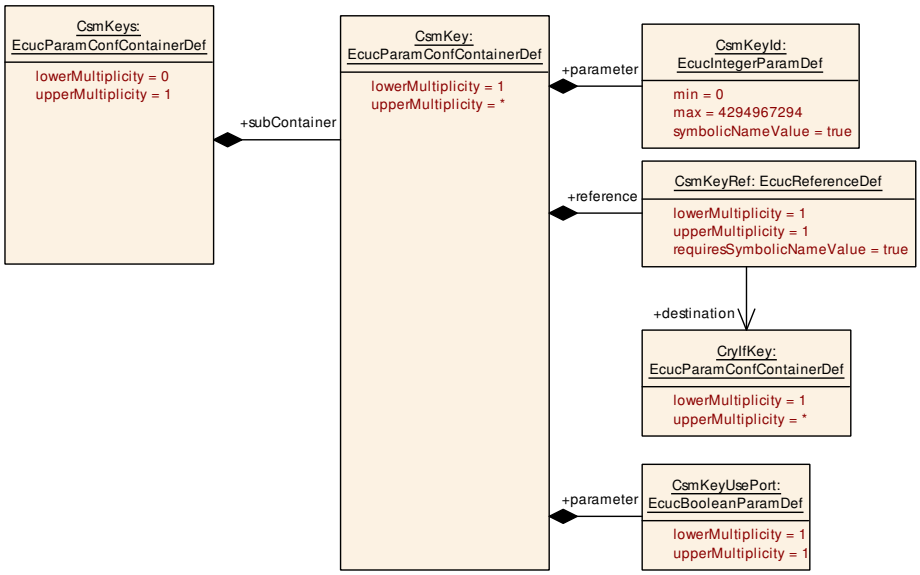


Figure 10.6: Crypto Service Manager Keys Layout

[ECUC_Csm_00005] Definition of EcucParamConfContainerDef CsmKeys ⌈

Container Name	CsmKeys
Parent Container	Csm
Upstream mapping defined	–

▽

△

Description	Container for CSM key configurations.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmKey	1..*	Container for configuration of a CSM key. The container name serves as a symbolic name for the identifier of a key configuration.

⌋

10.2.9 CsmKey

[ECUC_Csm_00014] Definition of EcucParamConfContainerDef CsmKey ⌈

Container Name	CsmKey
Parent Container	CsmKeys
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Container for configuration of a CSM key. The container name serves as a symbolic name for the identifier of a key configuration.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CsmKeyId	1	[ECUC_Csm_00015]
CsmKeyUsePort	1	[ECUC_Csm_00127]
CsmKeyRef	1	[ECUC_Csm_00016]

No Included Containers

⌋

[ECUC_Csm_00015] Definition of EcucIntegerParamDef CsmKeyId ⌈

Parameter Name	CsmKeyId
Parent Container	CsmKey
Upstream mapping defined	–

▽

△

Description	Identifier of the CsmKey. Dependency: The range is limited to 0xFFFFFFFE, to have the value 0xFFFFFFFF (CRYPTO_NO_ID) reserved for stack internal use. This is common for Crypto stack "Id" parameters.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 4294967294
Default value	–
Pre-Compile only	true

⌋

[ECUC_Csm_00127] Definition of EcucBooleanParamDef CsmKeyUsePort ⌈

Parameter Name	CsmKeyUsePort
Parent Container	CsmKey
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_Csm_00016] Definition of EcucReferenceDef CsmKeyRef ⌈

Parameter Name	CsmKeyRef
Parent Container	CsmKey
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

10.2.10 CsmPrimitives

[ECUC_Csm_00006] Definition of EcucParamConfContainerDef CsmPrimitives ⌈

Container Name	CsmPrimitives
Parent Container	Csm
Upstream mapping defined	–
Description	Container for configuration of CsmPrimitives

▽

△

Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmPrimitive	1..*	Container for configuration of a CSM primitive.

⌋

10.2.11 CsmPrimitive

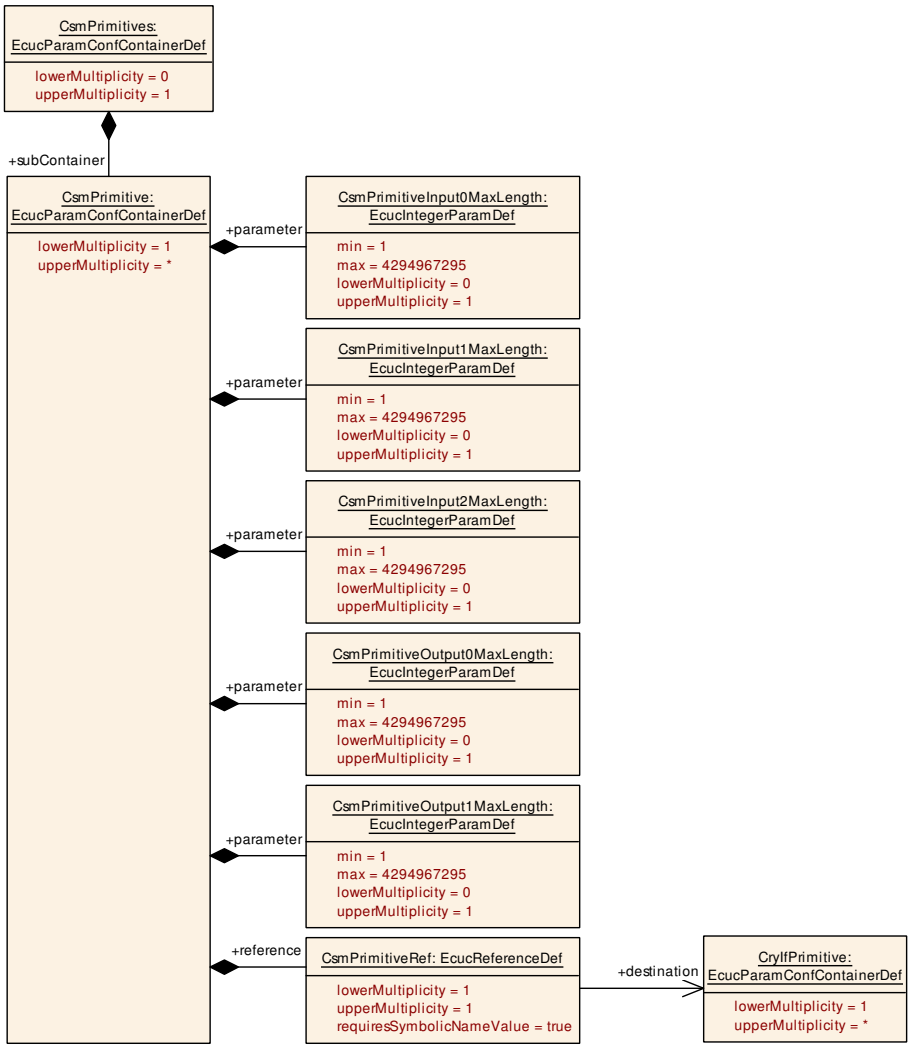


Figure 10.7: CsmPrimitives Layout

[ECUC_Csm_00391] Definition of EcucParamConfContainerDef CsmPrimitive ⌈

Container Name	CsmPrimitive
Parent Container	CsmPrimitives
Upstream mapping defined	–
Description	Container for configuration of a CSM primitive.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CsmPrimitiveInput0MaxLength	0..1	[ECUC_Csm_00392]
CsmPrimitiveInput1MaxLength	0..1	[ECUC_Csm_00393]
CsmPrimitiveInput2MaxLength	0..1	[ECUC_Csm_00394]
CsmPrimitiveOutput0MaxLength	0..1	[ECUC_Csm_00395]
CsmPrimitiveOutput1MaxLength	0..1	[ECUC_Csm_00396]
CsmPrimitiveRef	1	[ECUC_Csm_00400]

No Included Containers

⌋

[ECUC_Csm_00392] Definition of EcucIntegerParamDef CsmPrimitiveInput0MaxLength ⌈

Parameter Name	CsmPrimitiveInput0MaxLength	
Parent Container	CsmPrimitive	
Upstream mapping defined	–	
Description	Size of the input buffer (== job->action->jobInputsPtr[0]) 1. ciphertext (Decrypt, AEADDecrypt) 2. plaintext (Encrypt, AEADEncrypt) 3. data (Hash, MacGenerate, MacVerify, SignatureGenerate, SignatureVerify) for the synchronous RTE service interface if this primitive is referenced by a Csm job. It may also be possible that other BSW modules use the length parameter for buffer size calculation. Dependency: This parameter is only relevant if CsmJobInterfaceUsePort is configured as CRYPTO_USE_PORT.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00393] Definition of EcucIntegerParamDef CsmPrimitiveInput1MaxLength ⌈

Parameter Name	CsmPrimitiveInput1MaxLength	
Parent Container	CsmPrimitive	
Upstream mapping defined	–	
Description	Size of the input buffer (== job->action->jobInputsPtr[1]) 1. associated data (AEADDecrypt, AEADEncrypt) 2. MAC (MacVerify) 3. signature (SignatureVerify) for the synchronous RTE service interface if this primitive is referenced by a Csm job. It may also be possible that other BSW modules use the length parameter for buffer size calculation. Dependency: This parameter is only relevant if CsmJobInterfaceUsePort is configured as CRYPTO_USE_PORT.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00394] Definition of EcucIntegerParamDef CsmPrimitiveInput2MaxLength ⌈

Parameter Name	CsmPrimitiveInput2MaxLength	
Parent Container	CsmPrimitive	
Upstream mapping defined	–	
Description	Size of the input buffer (== job->action->jobInputsPtr[2]) 1. tag data (AEADDecrypt) for the synchronous RTE service interface if this primitive is referenced by a Csm job. It may also be possible that other BSW modules use the length parameter for buffer size calculation. Dependency: This parameter is only relevant if CsmJobInterfaceUsePort is configured as CRYPTO_USE_PORT.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00395] Definition of EcucIntegerParamDef CsmPrimitiveOutput0MaxLength ⌈

Parameter Name	CsmPrimitiveOutput0MaxLength
Parent Container	CsmPrimitive
Upstream mapping defined	–

▽

△

Description	Size of the output buffer (== job->action->jobOutputsPtr[0]) 1. plaintext (Decrypt, AEADDecrypt) 2. ciphertext (Encrypt, AEADEncrypt) 3. result (Hash, MacGenerate, RandomGenerate, SignatureGenerate) for the synchronous RTE service interface if this primitive is referenced by a Csm job. It may also be possible that other BSW modules use the length parameter for buffer size calculation. Dependency: This parameter is only relevant if CsmJobInterfaceUsePort is configured as CRYPTO_USE_PORT.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	1 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_Csm_00396] Definition of EcucIntegerParamDef CsmPrimitiveOutput1MaxLength ⌈

Parameter Name	CsmPrimitiveOutput1MaxLength
Parent Container	CsmPrimitive
Upstream mapping defined	–
Description	Size of the output buffer (== job->action->jobOutputsPtr[1]) 1. tag data (AEADEncrypt) for the synchronous RTE service interface if this primitive is referenced by a Csm job. It may also be possible that other BSW modules use the length parameter for buffer size calculation. Dependency: This parameter is only relevant if CsmJobInterfaceUsePort is configured as CRYPTO_USE_PORT.
Multiplicity	0..1
Type	EcucIntegerParamDef
Range	1 .. 4294967295
Default value	–
Pre-Compile only	true

⌋

[ECUC_Csm_00400] Definition of EcucReferenceDef CsmPrimitiveRef ⌈

Parameter Name	CsmPrimitiveRef
Parent Container	CsmPrimitive
Upstream mapping defined	–
Description	This parameter refers to the used CryIfPrimitive. The underlying CryIfPrimitive refers to a specific CryptoPrimitive in the Crypto Driver.
Multiplicity	1
Type	Symbolic name reference to CryIfPrimitive
Pre-Compile only	true

⌋

10.2.12 CsmCallbacks

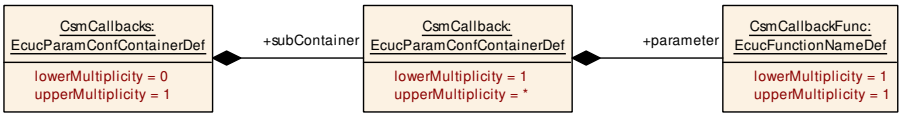


Figure 10.8: CsmCallbacks Layout

[ECUC_Csm_00008] Definition of EcucParamConfContainerDef CsmCallbacks ⌈

Container Name	CsmCallbacks
Parent Container	Csm
Upstream mapping defined	–
Description	Container for callback function configurations
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmCallback	1..*	Container for configuration of a callback function

⌋

10.2.13 CsmCallback

[ECUC_Csm_00109] Definition of EcucParamConfContainerDef CsmCallback ⌈

Container Name	CsmCallback
Parent Container	CsmCallbacks
Upstream mapping defined	–
Description	Container for configuration of a callback function
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CsmCallbackFunc	1	[ECUC_Csm_00110]

No Included Containers

⌋

[ECUC_Csm_00110] Definition of EcucFunctionNameDef CsmCallbackFunc ⌈

Parameter Name	CsmCallbackFunc
Parent Container	CsmCallback
Upstream mapping defined	–
Description	Callback function to be called if an asynchronous operation has finished. The corresponding job has to be configured to be processed asynchronously.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

10.2.14 CsmInputRedirectionSets

[ECUC_Csm_00377] Definition of EcucParamConfContainerDef CsmInputRedirectionSets ⌈

Container Name	CsmInputRedirectionSets
Parent Container	Csm
Upstream mapping defined	–
Description	Container for configuration of sets of CSM input redirections.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmInputRedirectionSet	1..*	Container for configuration of a single input redirection set.

⌋

10.2.15 CsmInputRedirectionSet

[ECUC_Csm_00381] Definition of EcucParamConfContainerDef CsmInputRedirectionSet ⌈

Container Name	CsmInputRedirectionSet
Parent Container	CsmInputRedirectionSets
Upstream mapping defined	–

▽

△

Description	Container for configuration of a single input redirection set.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
CsmInputRedirection	1..*	Container for configuration of a single input redirection.

⌋

10.2.16 CsmInputRedirection

[ECUC_Csm_00382] Definition of EcucParamConfContainerDef CsmInputRedirection ⌈

Container Name	CsmInputRedirection
Parent Container	CsmInputRedirectionSet
Upstream mapping defined	–
Description	Container for configuration of a single input redirection.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CsmInputRedirectionInputId	1	[ECUC_Csm_00383]
CsmInputRedirectionKeyElementId	1	[ECUC_Csm_00384]
CsmInputRedirectionKeyRef	1	[ECUC_Csm_00385]

No Included Containers

⌋

[ECUC_Csm_00383] Definition of EcucIntegerParamDef CsmInputRedirectionInputId ⌈

Parameter Name	CsmInputRedirectionInputId
Parent Container	CsmInputRedirection
Upstream mapping defined	–

▽

△

Description	The Index or Id of the Input that redirection refers to. If it's Input[0], then it's "0", etc. Dependency: The maximum value depends on CRYPTO_MAX_INPUTS (SWS_Csm_91116) and shall equal CRYPTO_MAX_INPUTS - 1. With CRYPTO_MAX_INPUTS currently defined as 3, the permitted range is 0 .. 2.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 2	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00384] Definition of EcucIntegerParamDef CsmInputRedirectionKeyElementId ⌈

Parameter Name	CsmInputRedirectionKeyElementId	
Parent Container	CsmInputRedirection	
Upstream mapping defined	–	
Description	Identifier of the key element used as input. Dependency: The range is limited to 0xFFFFFFFE, to have the value 0xFFFFFFFF (CRYPTO_NO_ID) reserved for stack internal use. This is common for Crypto stack "Id" parameters.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967294	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00385] Definition of EcucReferenceDef CsmInputRedirectionKeyRef ⌈

Parameter Name	CsmInputRedirectionKeyRef
Parent Container	CsmInputRedirection
Upstream mapping defined	–
Description	This parameter refers to the key used as input.
Multiplicity	1
Type	Reference to CsmKey
Pre-Compile only	true

⌋

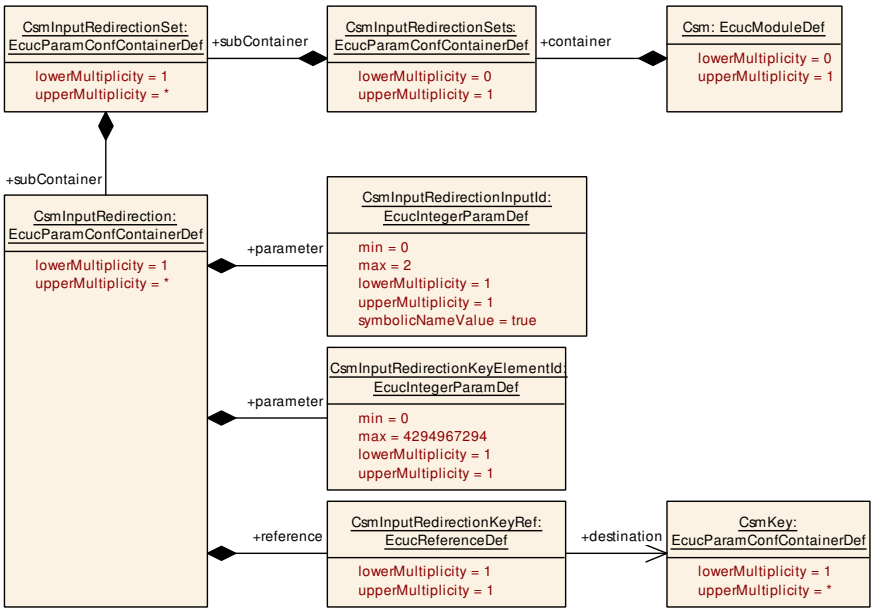


Figure 10.9: CsmInputRedirection Layout

10.2.17 CsmOutputRedirectionSets

[ECUC_Csm_00378] Definition of EcucParamConfContainerDef CsmOutputRedirectionSets ⌈

Container Name	CsmOutputRedirectionSets
Parent Container	Csm
Upstream mapping defined	–
Description	Container for configuration of sets of CSM output redirections.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
CsmOutputRedirectionSet	1..*	Container for configuration of a single output redirection set.

⌋

10.2.18 CsmOutputRedirectionSet

[ECUC_Csm_00386] Definition of EcucParamConfContainerDef CsmOutputRedirectionSet ⌈

Container Name	CsmOutputRedirectionSet
Parent Container	CsmOutputRedirectionSets
Upstream mapping defined	–
Description	Container for configuration of a single output redirection set.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	Multiplicity	Description
CsmOutputRedirection	1..*	Container for configuration of redirection respective to a single output.

⌋

10.2.19 CsmOutputRedirection

[ECUC_Csm_00387] Definition of EcucParamConfContainerDef CsmOutputRedirection ⌈

Container Name	CsmOutputRedirection
Parent Container	CsmOutputRedirectionSet
Upstream mapping defined	–
Description	Container for configuration of redirection respective to a single output.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
CsmOutputRedirectionKeyElementId	1	[ECUC_Csm_00389]
CsmOutputRedirectionOutputId	1	[ECUC_Csm_00388]
CsmOutputRedirectionKeyRef	1	[ECUC_Csm_00390]

No Included Containers

⌋

[ECUC_Csm_00389] Definition of EcucIntegerParamDef CsmOutputRedirection KeyElementId ⌈

Parameter Name	CsmOutputRedirectionKeyElementId	
Parent Container	CsmOutputRedirection	
Upstream mapping defined	–	
Description	Identifier of the key element used as output. Dependency: The range is limited to 0xFFFFFFFE, to have the value 0xFFFFFFFF (CRYPTO_NO_ID) reserved for stack internal use. This is common for Crypto stack "Id" parameters.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967294	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00388] Definition of EcucIntegerParamDef CsmOutputRedirection OutputId ⌈

Parameter Name	CsmOutputRedirectionOutputId	
Parent Container	CsmOutputRedirection	
Upstream mapping defined	–	
Description	The Index or Id of the Output that redirection refers to. If it's Output[0], then it's "0", etc. Dependency: The maximum value depends on CRYPTO_MAX_OUTPUTS (SWS_Csm_91117) and shall equal CRYPTO_MAX_OUTPUTS - 1. With CRYPTO_MAX_OUTPUTS currently defined as 2, the permitted range is 0 .. 1.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 1	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_Csm_00390] Definition of EcucReferenceDef CsmOutputRedirectionKey Ref ⌈

Parameter Name	CsmOutputRedirectionKeyRef
Parent Container	CsmOutputRedirection
Upstream mapping defined	–
Description	This parameter refers to the key used as output.
Multiplicity	1
Type	Reference to CsmKey
Pre-Compile only	true

⌋

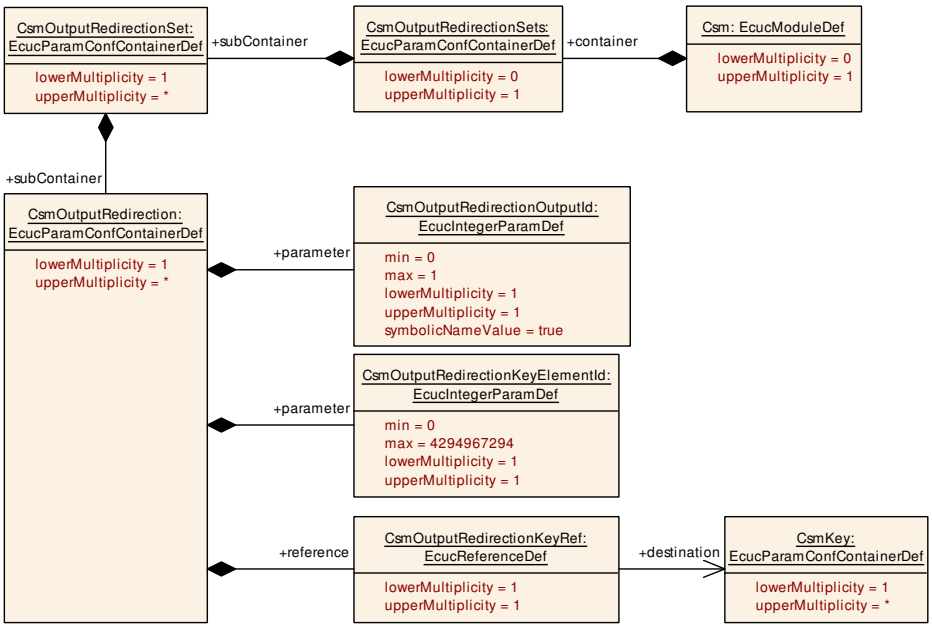


Figure 10.10: CsmOutputRedirection Layout

10.3 Published Information

For details refer to [5] Chapter 10.3 *“Published Information”*.

A Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

A.1 Traceable item history of this document according to AUTOSAR Release R25-11

A.1.1 Added Specification Items in R25-11

[SWS_Csm_00960]

A.1.2 Changed Specification Items in R25-11

[SWS_Csm_00952]

A.1.3 Deleted Specification Items in R25-11

none

A.2 Traceable item history of this document according to AUTOSAR Release R26-11

A.2.1 Added Specification Items in R26-11

[ECUC_Csm_00376] [ECUC_Csm_00377] [ECUC_Csm_00378] [ECUC_Csm_00379] [ECUC_Csm_00380] [ECUC_Csm_00381] [ECUC_Csm_00382] [ECUC_Csm_00383] [ECUC_Csm_00384] [ECUC_Csm_00385] [ECUC_Csm_00386] [ECUC_Csm_00387] [ECUC_Csm_00388] [ECUC_Csm_00389] [ECUC_Csm_00390] [ECUC_Csm_00391] [ECUC_Csm_00392] [ECUC_Csm_00393] [ECUC_Csm_00394] [ECUC_Csm_00395] [ECUC_Csm_00396] [ECUC_Csm_00397] [ECUC_Csm_00398] [ECUC_Csm_00399] [ECUC_Csm_00400] [SWS_Csm_91076] [SWS_Csm_91077] [SWS_Csm_91078] [SWS_Csm_91079] [SWS_Csm_91080] [SWS_Csm_91081] [SWS_Csm_91082] [SWS_Csm_91083] [SWS_Csm_91084] [SWS_Csm_91085] [SWS_Csm_91086] [SWS_Csm_91087] [SWS_Csm_91088] [SWS_Csm_91089] [SWS_Csm_91090] [SWS_Csm_91091] [SWS_Csm_91092] [SWS_Csm_91093] [SWS_Csm_91094] [SWS_Csm_91095] [SWS_Csm_91096] [SWS_Csm_91116] [SWS_Csm_91117] [SWS_Csm_91118] [SWS_Csm_91119] [SWS_Csm_91120] [SWS_Csm_91121] [SWS_Csm_91122] [SWS_Csm_91123]

[SWS_Csm_91124] [SWS_Csm_91125] [SWS_Csm_91126] [SWS_Csm_91127] [SWS_Csm_91128]

A.2.2 Changed Specification Items in R26-11

[ECUC_Csm_00002] [ECUC_Csm_00005] [ECUC_Csm_00006] [ECUC_Csm_00008] [ECUC_Csm_00015] [ECUC_Csm_00033] [ECUC_Csm_00118] [ECUC_Csm_00119] [ECUC_Csm_00122] [ECUC_Csm_00126] [ECUC_Csm_00818] [SWS_Csm_00016] [SWS_Csm_00017] [SWS_Csm_00032] [SWS_Csm_00902] [SWS_Csm_00903] [SWS_Csm_00922] [SWS_Csm_00936] [SWS_Csm_00941] [SWS_Csm_00943] [SWS_Csm_00946] [SWS_Csm_00947] [SWS_Csm_01021] [SWS_Csm_01031] [SWS_Csm_01039] [SWS_Csm_01044] [SWS_Csm_01049] [SWS_Csm_01074] [SWS_Csm_01075] [SWS_Csm_01096] [SWS_Csm_01906] [SWS_Csm_01910] [SWS_Csm_01915] [SWS_Csm_01925] [SWS_Csm_01926] [SWS_Csm_01927] [SWS_Csm_09000] [SWS_Csm_91004] [SWS_Csm_91008] [SWS_Csm_91014] [SWS_Csm_91017] [SWS_Csm_91023] [SWS_Csm_91067] [SWS_Csm_91068] [SWS_Csm_91069]

A.2.3 Deleted Specification Items in R26-11

[ECUC_Csm_00003] [ECUC_Csm_00007] [ECUC_Csm_00021] [ECUC_Csm_00022] [ECUC_Csm_00023] [ECUC_Csm_00024] [ECUC_Csm_00025] [ECUC_Csm_00026] [ECUC_Csm_00027] [ECUC_Csm_00028] [ECUC_Csm_00029] [ECUC_Csm_00031] [ECUC_Csm_00032] [ECUC_Csm_00034] [ECUC_Csm_00036] [ECUC_Csm_00038] [ECUC_Csm_00040] [ECUC_Csm_00041] [ECUC_Csm_00044] [ECUC_Csm_00049] [ECUC_Csm_00051] [ECUC_Csm_00056] [ECUC_Csm_00057] [ECUC_Csm_00060] [ECUC_Csm_00064] [ECUC_Csm_00066] [ECUC_Csm_00067] [ECUC_Csm_00068] [ECUC_Csm_00072] [ECUC_Csm_00074] [ECUC_Csm_00075] [ECUC_Csm_00076] [ECUC_Csm_00080] [ECUC_Csm_00081] [ECUC_Csm_00082] [ECUC_Csm_00083] [ECUC_Csm_00084] [ECUC_Csm_00086] [ECUC_Csm_00087] [ECUC_Csm_00089] [ECUC_Csm_00090] [ECUC_Csm_00091] [ECUC_Csm_00094] [ECUC_Csm_00096] [ECUC_Csm_00098] [ECUC_Csm_00103] [ECUC_Csm_00105] [ECUC_Csm_00106] [ECUC_Csm_00107] [ECUC_Csm_00125] [ECUC_Csm_00130] [ECUC_Csm_00131] [ECUC_Csm_00134] [ECUC_Csm_00137] [ECUC_Csm_00138] [ECUC_Csm_00140] [ECUC_Csm_00142] [ECUC_Csm_00144] [ECUC_Csm_00146] [ECUC_Csm_00147] [ECUC_Csm_00149] [ECUC_Csm_00154] [ECUC_Csm_00155] [ECUC_Csm_00156] [ECUC_Csm_00157] [ECUC_Csm_00158] [ECUC_Csm_00159] [ECUC_Csm_00160] [ECUC_Csm_00161] [ECUC_Csm_00162] [ECUC_Csm_00163] [ECUC_Csm_00164] [ECUC_Csm_00165] [ECUC_Csm_00169] [ECUC_Csm_00170] [ECUC_Csm_00172] [ECUC_Csm_00175] [ECUC_Csm_00176] [ECUC_Csm_00178] [ECUC_Csm_00181] [ECUC_Csm_00182] [ECUC_Csm_00183] [ECUC_Csm_00188] [ECUC_Csm_00189] [ECUC_Csm_00191] [ECUC_Csm_00192] [ECUC_Csm_00193] [ECUC_Csm_00195]

[ECUC_Csm_00196] [ECUC_Csm_00197] [ECUC_Csm_00198] [ECUC_Csm_00199] [ECUC_Csm_00200] [ECUC_Csm_00201] [ECUC_Csm_00204] [ECUC_Csm_00206] [ECUC_Csm_00208] [ECUC_Csm_00210] [ECUC_Csm_00213] [ECUC_Csm_00215] [ECUC_Csm_00216] [ECUC_Csm_00218] [ECUC_Csm_00220] [ECUC_Csm_00222] [ECUC_Csm_00223] [ECUC_Csm_00225] [ECUC_Csm_00226] [ECUC_Csm_00227] [ECUC_Csm_00229] [ECUC_Csm_00231] [ECUC_Csm_00234] [ECUC_Csm_00235] [ECUC_Csm_00237] [ECUC_Csm_00239] [ECUC_Csm_00261] [ECUC_Csm_00262] [ECUC_Csm_00263] [ECUC_Csm_00264] [ECUC_Csm_00265] [ECUC_Csm_00266] [ECUC_Csm_00267] [ECUC_Csm_00268] [ECUC_Csm_00269] [ECUC_Csm_00270] [ECUC_Csm_00271] [ECUC_Csm_00272] [ECUC_Csm_00273] [ECUC_Csm_00274] [ECUC_Csm_00277] [ECUC_Csm_00278] [ECUC_Csm_00281] [ECUC_Csm_00282] [ECUC_Csm_00283] [ECUC_Csm_00284] [ECUC_Csm_00285] [ECUC_Csm_00286] [ECUC_Csm_00287] [ECUC_Csm_00288] [ECUC_Csm_00289] [ECUC_Csm_00290] [ECUC_Csm_00291] [ECUC_Csm_00292] [ECUC_Csm_00293] [ECUC_Csm_00294] [ECUC_Csm_00295] [ECUC_Csm_00296] [ECUC_Csm_00297] [ECUC_Csm_00298] [ECUC_Csm_00299] [ECUC_Csm_00300] [ECUC_Csm_00301] [ECUC_Csm_00302] [ECUC_Csm_00303] [ECUC_Csm_00304] [ECUC_Csm_00305] [ECUC_Csm_00306] [ECUC_Csm_00307] [ECUC_Csm_00308] [ECUC_Csm_00309] [ECUC_Csm_00310] [ECUC_Csm_00311] [ECUC_Csm_00312] [ECUC_Csm_00313] [ECUC_Csm_00314] [ECUC_Csm_00315] [ECUC_Csm_00316] [ECUC_Csm_00317] [ECUC_Csm_00318] [ECUC_Csm_00319] [ECUC_Csm_00320] [ECUC_Csm_00321] [ECUC_Csm_00322] [ECUC_Csm_00323] [ECUC_Csm_00324] [ECUC_Csm_00325] [ECUC_Csm_00326] [ECUC_Csm_00328] [ECUC_Csm_00329] [ECUC_Csm_00330] [ECUC_Csm_00331] [ECUC_Csm_00332] [ECUC_Csm_00333] [ECUC_Csm_00334] [ECUC_Csm_00335] [ECUC_Csm_00336] [ECUC_Csm_00337] [ECUC_Csm_00338] [ECUC_Csm_00339] [ECUC_Csm_00340] [ECUC_Csm_00341] [ECUC_Csm_00342] [ECUC_Csm_00343] [ECUC_Csm_00344] [ECUC_Csm_00345] [ECUC_Csm_00346] [ECUC_Csm_00347] [ECUC_Csm_00348] [ECUC_Csm_00349] [ECUC_Csm_00350] [ECUC_Csm_00351] [ECUC_Csm_00352] [ECUC_Csm_00353] [ECUC_Csm_00354] [ECUC_Csm_00355] [ECUC_Csm_00356] [ECUC_Csm_00357] [ECUC_Csm_00358] [ECUC_Csm_00359] [ECUC_Csm_00360] [ECUC_Csm_00361] [ECUC_Csm_00362] [ECUC_Csm_00363] [ECUC_Csm_00364] [ECUC_Csm_00365] [ECUC_Csm_00367] [ECUC_Csm_00369] [ECUC_Csm_00370] [ECUC_Csm_00371] [ECUC_Csm_00372] [ECUC_Csm_00373] [ECUC_Csm_00374] [ECUC_Csm_00375] [SWS_Csm_00018] [SWS_Csm_00019] [SWS_Csm_00022] [SWS_Csm_00028] [SWS_Csm_00029] [SWS_Csm_00035] [SWS_Csm_00036] [SWS_Csm_00037] [SWS_Csm_00039] [SWS_Csm_00705] [SWS_Csm_00802] [SWS_Csm_00803] [SWS_Csm_00912] [SWS_Csm_00923] [SWS_Csm_00927] [SWS_Csm_00930] [SWS_Csm_00935] [SWS_Csm_00940] [SWS_Csm_00942] [SWS_Csm_00944] [SWS_Csm_00974] [SWS_Csm_01008] [SWS_Csm_01009] [SWS_Csm_01011] [SWS_Csm_01012] [SWS_Csm_01013] [SWS_Csm_01028] [SWS_Csm_01030] [SWS_Csm_01041] [SWS_Csm_01047] [SWS_Csm_01048] [SWS_Csm_01077] [SWS_Csm_01078] [SWS_Csm_01079] [SWS_Csm_01083] [SWS_Csm_01086] [SWS_Csm_01087] [SWS_Csm_01088]

[SWS_Csm_01089] [SWS_Csm_01093] [SWS_Csm_01094] [SWS_Csm_01920] [SWS_Csm_01921] [SWS_Csm_01922] [SWS_Csm_01923] [SWS_Csm_01924] [SWS_Csm_01928] [SWS_Csm_91005] [SWS_Csm_91006] [SWS_Csm_91007] [SWS_Csm_91015] [SWS_Csm_91016] [SWS_Csm_91024] [SWS_Csm_91026] [SWS_Csm_91065] [SWS_Csm_91072] [SWS_Csm_91073] [SWS_Csm_91074]