Document Title	Specification of Crypto Service Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	402
Document Classification	Standard
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
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

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
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

Advice for users

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1 Introduction and Functional Overview

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

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

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

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

2 Acronyms and Abbreviations

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

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

2.1 Glossary of Terms

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

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

3 Related documentation

3.1 Input Documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[4] Specification of RTE Software
AUTOSAR_SWS_RTE.pdf

[5] Specification of BSW Scheduler
AUTOSAR_SWS_Scheduler.pdf

[6] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[7] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

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

[9] Specification of Diagnostic Event Manager
AUTOSAR_SWS_DiagnosticEventManager.pdf

[10] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[11] Specification of C Implementation Rules
AUTOSAR_TR_CImplementationRules.pdf

[12] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[13] AUTOSAR Glossary
AUTOSAR_TR_Glossary.pdf

[14] Requirements on the Crypto Stack
AUTOSAR_SRS_CryptoStack.pdf

[15] Specification of the Crypto Interface
AUTOSAR_SWS_CryptoInterface.pdf
[16] Specification of the Crypto Driver

AUTOSAR_SWS_CryptoDriver.pdf

[17] Specification of Crypto Abstraction Library
AUTOSAR_SWS_CryptoAbstractionLibrary.pdf

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

3.2 Related standards and norms

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

3.3 Related specification

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

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

4 Constraints and Assumptions

4.1 Limitations

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

4.2 Applicability to Car Domains

n.a.

4.3 Security Implications

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

5 Dependencies to other Modules

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

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

5.1 File Structure

5.1.1 Code File Structure

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

5.1.2 Header File Structure

[SWS_Csm_00005] ⌈The header file structure shall contain an application interface header file Csm.h, that provides the function prototypes to access the CSM services.
⌋()

[SWS_Csm_00003] ⌈The header file structure shall contain a configuration header Csm_Cfg.h, that provides the configuration parameters for the CSM module.
⌋(SRS_BSW_00345)

[SWS_Csm_00727] ⌈The header file structure shall contain a callback interface Csm_Cbk.h, that provides the callback function prototypes.
⌋(SRS_BSW_00346)

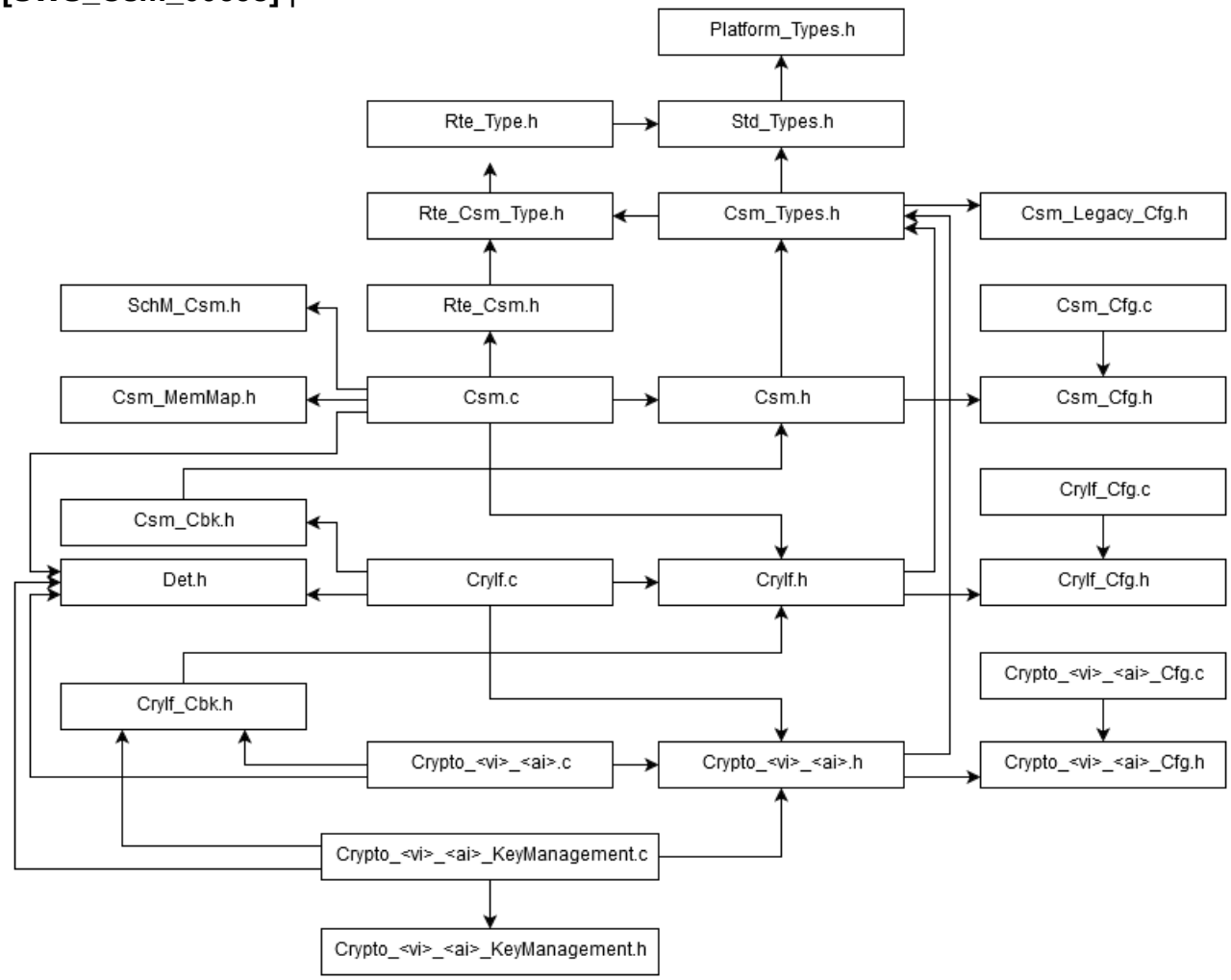
[SWS_Csm_00004] ⌈The header file structure shall contain a type header Csm_Types.h, that provides the types, particularly configuration types, for the CSM module. This file shall only contain types, that are not already defined in Rte_Csm_Type.h.
⌋()

[SWS_Csm_00006] ⌈The header file structure shall contain a legacy config header Csm_Legacy_Cfg.h, that provides configurations needed for the legacy API.
⌋()

[SWS_Csm_00694] ⌈The underlying CRYIF module shall provide a header file CryIf.h.
⌋()

[SWS_Csm_00008] ⌈The Figure in SWS_Csm_00695 (CSM File Structure) shows the include file structure.
⌋(SRS_BSW_00348)

[SWS_Csm_00695] ⌈



⌋(SRS_BSW_00348)

6 Requirements Traceability

Requirement	Description	Satisfied by
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_Csm_00646
SRS_BSW_00337	Classification of development errors	SWS_Csm_00489, SWS_Csm_00539
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_Csm_00003
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_Csm_00727
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Csm_00008, SWS_Csm_00695
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Csm_00646
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Csm_00073, SWS_Csm_00455, SWS_Csm_00457, SWS_Csm_00970
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Csm_00073, SWS_Csm_00455, SWS_Csm_00457, SWS_Csm_00970
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_00385	List possible error notifications	SWS_Csm_00539
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_Csm_00489, SWS_Csm_00539
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Csm_00705
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_Csm_00646
SRS_BSW_00432	Modules should have separate main processing	SWS_Csm_00479

	functions for read/receive and write/transmit data path	
SRS_CryptoStack_00008	The Crypto Stack shall allow static configuration of keys used for cryptographic jobs	SWS_Csm_01012
SRS_CryptoStack_00009	The Crypto Stack shall support reentrancy for all crypto services	SWS_Csm_00022
SRS_CryptoStack_00010	The Crypto Stack shall conceal symmetric keys from the users of crypto services	SWS_Csm_00959
SRS_CryptoStack_00011	The Crypto Stack shall conceal asymmetric private keys from the users of Crypto services	SWS_Csm_00959
SRS_CryptoStack_00019	The Crypto Stack shall identify random number generation as a cryptographic primitive which can be requested to a driver	SWS_Csm_01543
SRS_CryptoStack_00020	The Crypto Stack shall identify symmetric encryption/decryption as a cryptographic primitive which can be requested to a driver	SWS_Csm_00984, SWS_Csm_00989
SRS_CryptoStack_00021	The Crypto Stack shall identify asymmetric encryption/decryption as a cryptographic primitive which can be requested to a driver	SWS_Csm_00984, SWS_Csm_00989
SRS_CryptoStack_00022	The Crypto Stack shall identify MAC generation/verification as a cryptographic primitive which can be requested to a driver	SWS_Csm_00982
SRS_CryptoStack_00023	The Crypto Stack shall identify asymmetric signature generation/verification as a cryptographic primitive which can be requested to a driver	SWS_Csm_00992, SWS_Csm_00996
SRS_CryptoStack_00024	The Crypto Stack shall identify hash calculation as a cryptographic primitive which can be requested to a driver	SWS_Csm_00980
SRS_CryptoStack_00026	The Crypto Stack shall provide an interface for the generation of asymmetric keys	SWS_Csm_00955
SRS_CryptoStack_00027	The Crypto Stack shall provide an interface for the generation of symmetric keys	SWS_Csm_00955
SRS_CryptoStack_00082	The CSM module specification shall specify the	SWS_Csm_00001, SWS_Csm_00032, SWS_Csm_00506

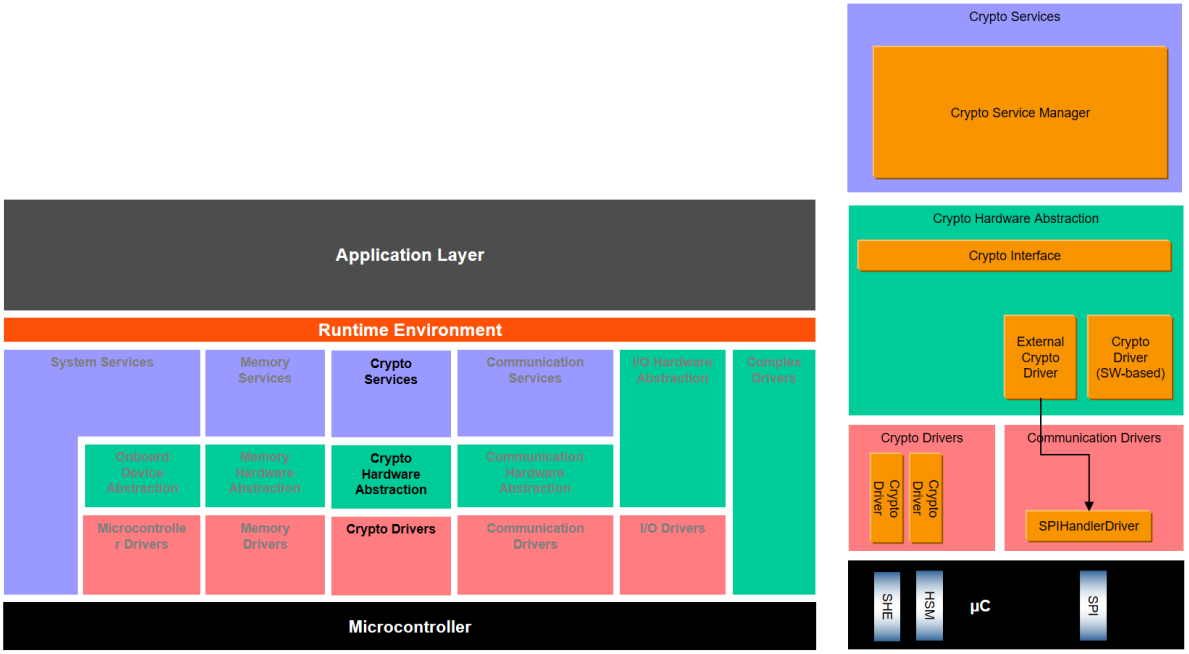
	interface and behavior of the callback function, if the asynchronous job processing mode is selected	
SRS_CryptoStack_00084	The CSM module shall use the streaming approach for some selected services	SWS_Csm_01039
SRS_CryptoStack_00086	The CSM module shall distinguish between error types	SWS_Csm_00828
SRS_CryptoStack_00087	The CSM module shall report detected development errors to the Development Error Tracer	SWS_Csm_00489, SWS_Csm_00539, SWS_Csm_00830
SRS_CryptoStack_00088	The CSM shall check passed API parameters for validity	SWS_Csm_00489, SWS_Csm_00539
SRS_CryptoStack_00090	The CSM shall provide an interface to be accessible via the RTE	SWS_Csm_00073, SWS_Csm_00076, SWS_Csm_00077, SWS_Csm_00081, SWS_Csm_00083, SWS_Csm_00802, SWS_Csm_00803, SWS_Csm_00825, SWS_Csm_00827, SWS_Csm_00829, SWS_Csm_00832, SWS_Csm_00833, SWS_Csm_00834, SWS_Csm_00835, SWS_Csm_00837, SWS_Csm_00838, SWS_Csm_009000, SWS_Csm_00902, SWS_Csm_00903, SWS_Csm_00912, SWS_Csm_00922, SWS_Csm_00923, SWS_Csm_00927, SWS_Csm_00928, SWS_Csm_00930, SWS_Csm_00931, SWS_Csm_00932, SWS_Csm_00933, SWS_Csm_00934, SWS_Csm_00935, SWS_Csm_00936, SWS_Csm_00943, SWS_Csm_00946, SWS_Csm_01042, SWS_Csm_01074, SWS_Csm_01075, SWS_Csm_01077, SWS_Csm_01078, SWS_Csm_01079, SWS_Csm_01080, SWS_Csm_01082, SWS_Csm_01906, SWS_Csm_01910, SWS_Csm_01915, SWS_Csm_01920, SWS_Csm_01921, SWS_Csm_01922, SWS_Csm_01923, SWS_Csm_01924, SWS_Csm_01925, SWS_Csm_01926, SWS_Csm_01927, SWS_Csm_01928, SWS_Csm_09260
SRS_CryptoStack_00091	The CSM shall provide one Provide--Port for each configuration	SWS_Csm_00825, SWS_Csm_00832, SWS_Csm_00833, SWS_Csm_00834, SWS_Csm_00835, SWS_Csm_00837, SWS_Csm_00838, SWS_Csm_00931, SWS_Csm_00932, SWS_Csm_00933, SWS_Csm_00934, SWS_Csm_01042
SRS_CryptoStack_00095	The Crypto Driver module shall strictly separate error and status information	SWS_Csm_01069, SWS_Csm_91001
SRS_CryptoStack_00100	Synchronous Job Processing	SWS_Csm_01049
SRS_CryptoStack_00101	Asynchronous Job Processing	SWS_Csm_01049
SRS_CryptoStack_00102	The priority of a user and its	SWS_Csm_01010

	crypto jobs shall be defined by static configuration	
SRS_CryptoStack_00103	The Crypto Stack shall provide an interface for the derivation of symmetric keys	SWS_Csm_00956
SRS_CryptoStack_00906	-	SWS_Csm_00947
SRS_CryptoStack_01076	-	SWS_Csm_01083
SRS_CrytptoStack_00028	-	SWS_Csm_00966, SWS_Csm_00967
SRS_CrytptoStack_00029	-	SWS_Csm_00959
SRS_CrytptoStack_00030	-	SWS_Csm_00998, SWS_Csm_00999
SRS_CrytptoStack_00031	-	SWS_Csm_01036
SRS_Csm_00046	-	SWS_Csm_00069
SRS_Csm_00066	-	SWS_Csm_00075, SWS_Csm_00086, SWS_Csm_00087, SWS_Csm_00691, SWS_Csm_00728, SWS_Csm_00775, SWS_Csm_00776, SWS_Csm_00777, SWS_Csm_00778, SWS_Csm_00779, SWS_Csm_0078, SWS_Csm_00780, SWS_Csm_00781, SWS_Csm_00782, SWS_Csm_00783, SWS_Csm_00784, SWS_Csm_00785, SWS_Csm_00786, SWS_Csm_00787, SWS_Csm_00789, SWS_Csm_00790, SWS_Csm_00791, SWS_Csm_00792, SWS_Csm_00793, SWS_Csm_00794, SWS_Csm_00795, SWS_Csm_00796, SWS_Csm_00797, SWS_Csm_00798, SWS_Csm_00799, SWS_Csm_00800, SWS_Csm_00801, SWS_Csm_00805, SWS_Csm_00812, SWS_Csm_00813, SWS_Csm_00817, SWS_Csm_00821, SWS_Csm_00824, SWS_Csm_00825, SWS_Csm_00840, SWS_Csm_00841, SWS_Csm_00842, SWS_Csm_00843, SWS_Csm_00845, SWS_Csm_00846, SWS_Csm_00847, SWS_Csm_00848, SWS_Csm_00849, SWS_Csm_00850, SWS_Csm_00851, SWS_Csm_00852, SWS_Csm_00853, SWS_Csm_00854, SWS_Csm_00855, SWS_Csm_00856, SWS_Csm_00857, SWS_Csm_00858, SWS_Csm_00859, SWS_Csm_00860, SWS_Csm_00861, SWS_Csm_00862, SWS_Csm_00863, SWS_Csm_00864, SWS_Csm_00865, SWS_Csm_00866, SWS_Csm_00867, SWS_Csm_00868, SWS_Csm_00869, SWS_Csm_00870, SWS_Csm_00871, SWS_Csm_00872, SWS_Csm_00873, SWS_Csm_00874, SWS_Csm_00875, SWS_Csm_00876, SWS_Csm_00877, SWS_Csm_00878, SWS_Csm_00879, SWS_Csm_00880, SWS_Csm_00881, SWS_Csm_00882, SWS_Csm_00883, SWS_Csm_00884, SWS_Csm_00885, SWS_Csm_00886, SWS_Csm_00887,

		SWS_Csm_00888, SWS_Csm_00889, SWS_Csm_00890, SWS_Csm_00891, SWS_Csm_00892, SWS_Csm_00893, SWS_Csm_00894, SWS_Csm_00895, SWS_Csm_00896, SWS_Csm_00897, SWS_Csm_00898, SWS_Csm_00899, SWS_Csm_00901, SWS_Csm_00902, SWS_Csm_00903, SWS_Csm_00904, SWS_Csm_00905, SWS_Csm_00906, SWS_Csm_00907, SWS_Csm_00908, SWS_Csm_00909, SWS_Csm_00910, SWS_Csm_00911, SWS_Csm_00913, SWS_Csm_00914, SWS_Csm_00915, SWS_Csm_00916, SWS_Csm_00917, SWS_Csm_00918, SWS_Csm_00919, SWS_Csm_00920, SWS_Csm_00921, SWS_Csm_01905

7 Functional specification

AUTOSAR Layered View [2].



AUTOSAR Layered View with CSM

7.1 Basic Architecture Guidelines

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

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

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

7.2 General Behavior

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

⌋()

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

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

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

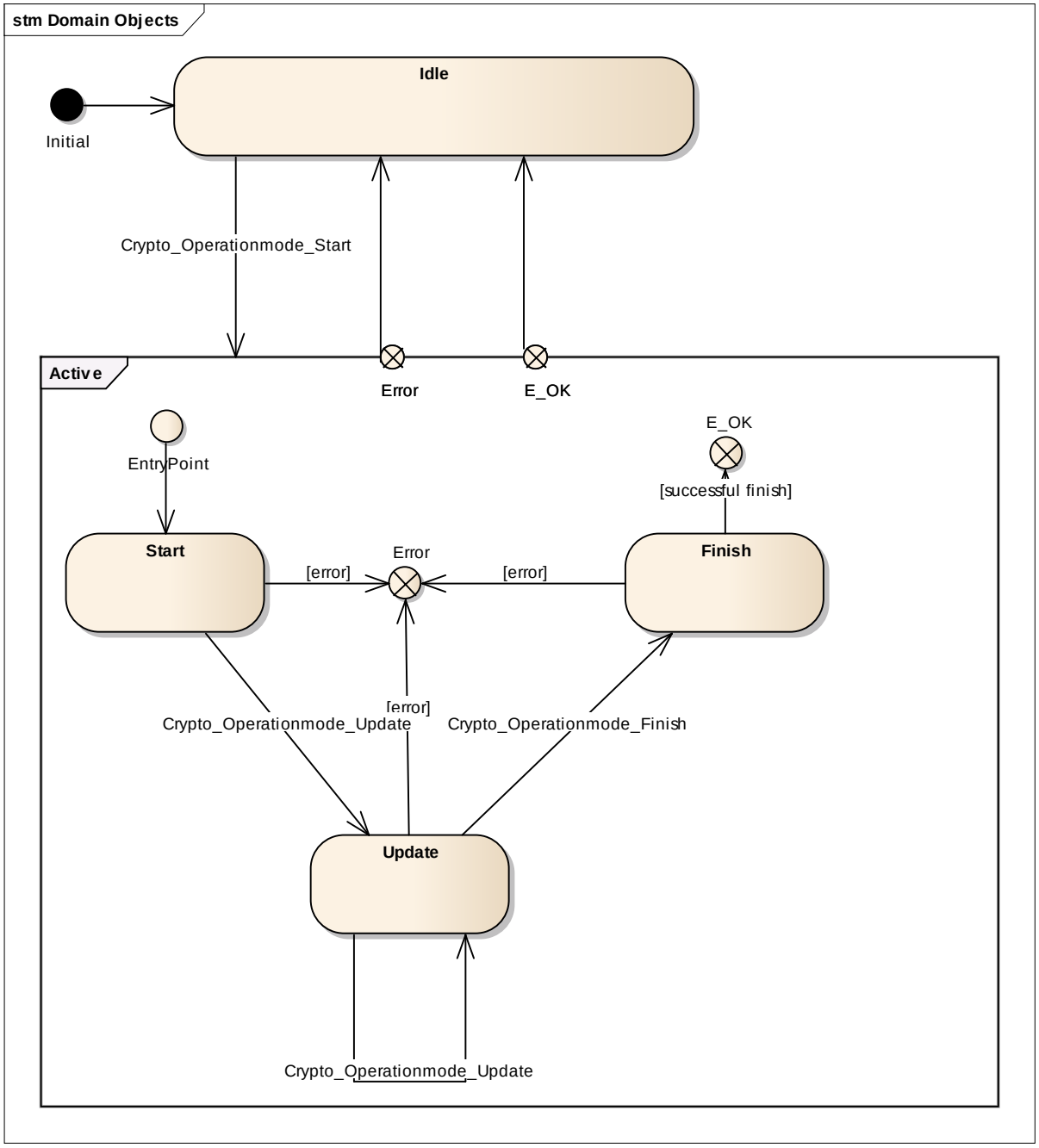
[SWS_Csm_00019] ⌈If an asynchronous interface is configured, the CSM module shall provide a main function Csm_MainFunction() which is called cyclically to control processing of the jobs via a state machine.
⌋()

7.2.1 Normal Operation

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

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

[SWS_Csm_00024] ⌈



](

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

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

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

7.2.1.1 Configuration

[SWS_Csm_00929] ⌈Each job configuration shall be realized as a constant structure of type `Crypto_JobType`.
⌋()

[SWS_Csm_00930] ⌈Each crypto primitive configuration shall be realized as a constant structure of type `Crypto_PrimitiveInfoType`.
⌋()

[SWS_Csm_00932] ⌈Each job primitive configuration shall be realized as a constant structure of type `Crypto_JobPrimitiveInfoType`.
⌋()

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

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

7.2.1.2 Synchronous Job Processing

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

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

[SWS_Csm_00037] [If a synchronous job is issued and the priority is less than the highest priority available in the queue, the CSM shall return E_BUSY.
](()

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

7.2.1.3 Asynchronous Job Processing

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

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

7.2.2 Design Notes

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

7.2.2.1 CSM module startup

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

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

7.2.2.2 Crypto Services

7.2.2.2.1 Usage of the CSM crypto services

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

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

[SWS_Csm_00925]⌈The application shall be able to call arbitrary often `Csm_<Service>()` with the operation mode `CRYPTO_OPERATIONMODE_UPDATE` 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.
⌋()

DEPRECATED: The Start/Update/Finish functions will be removed in the next major release!

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

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

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

7.2.2.2.2 Queuing

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

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

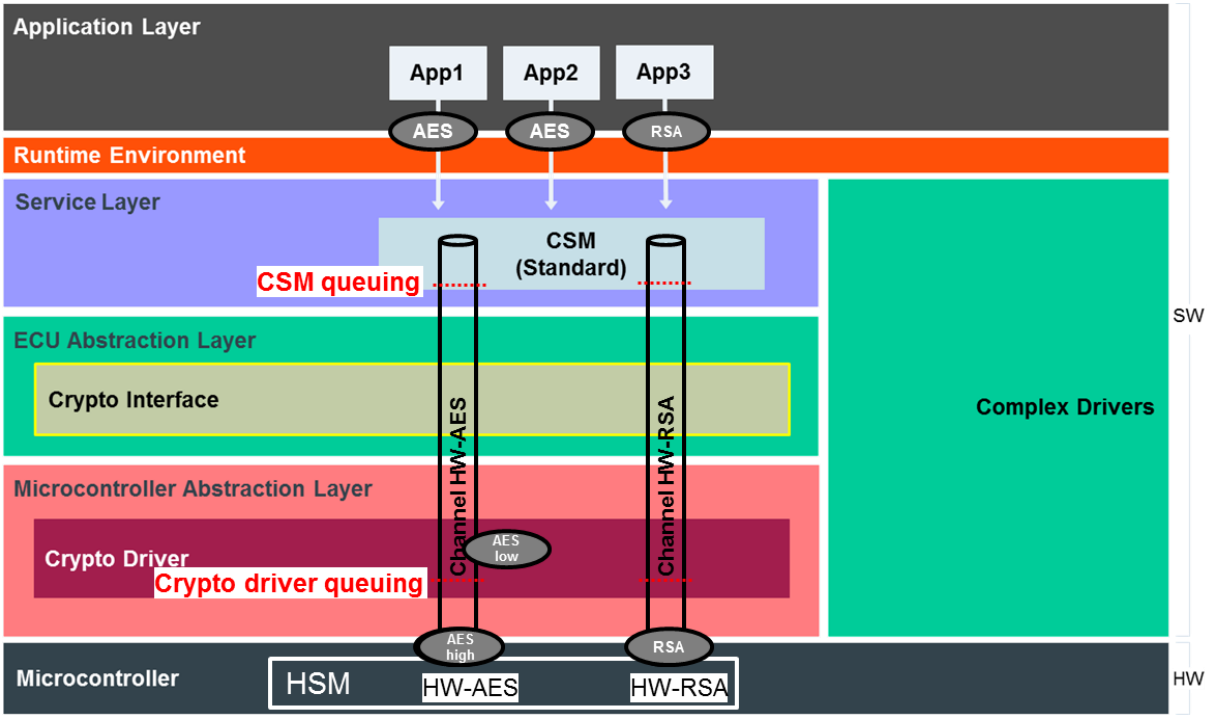


Figure 7.1 AUTOSAR Layered View with channels

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

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

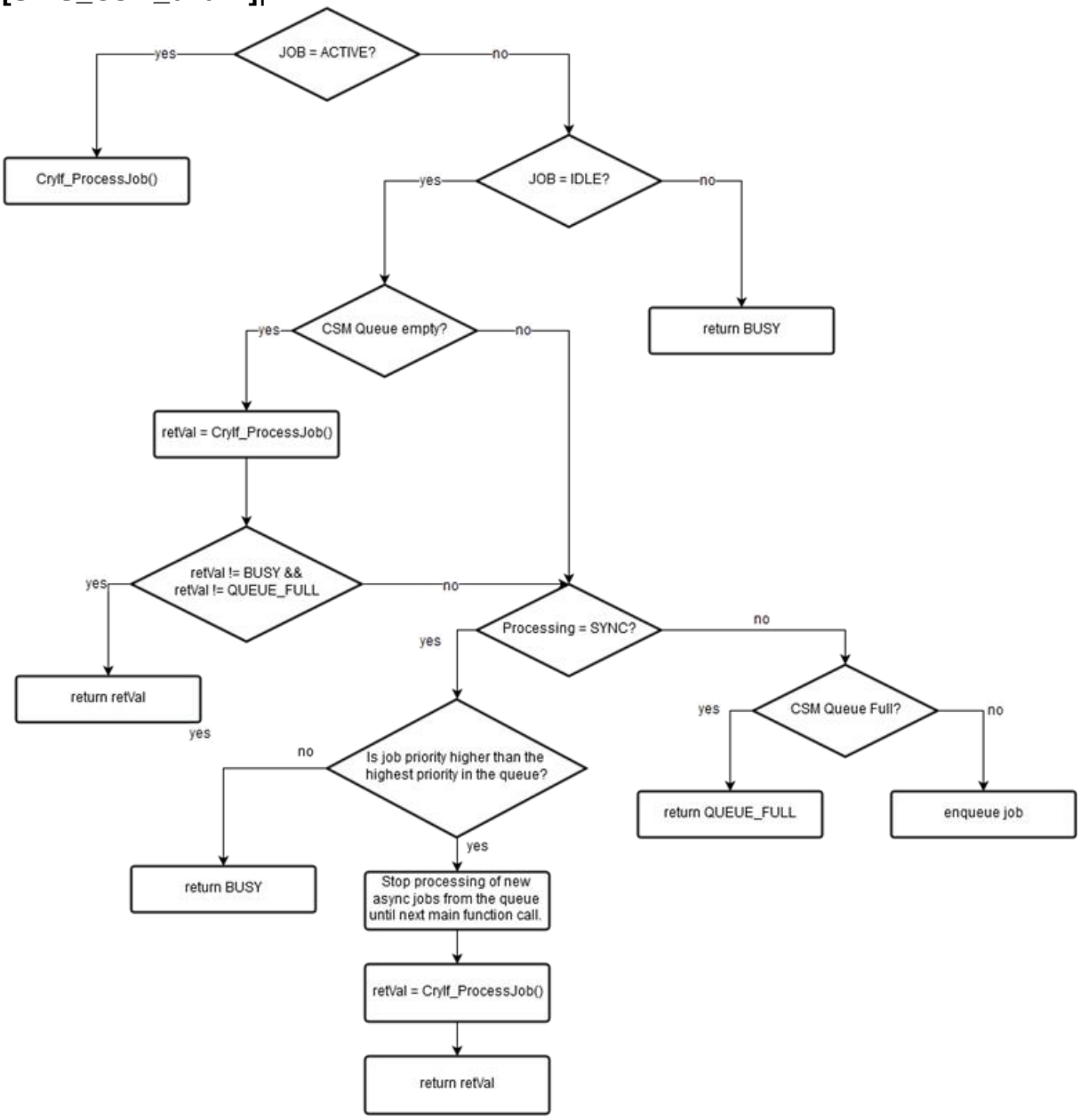
- If the Crypto Driver Object of the RSA is not busy, the job will be processed immediately.
- If the Crypto Driver Object of the RSA is busy, but the queue of the Crypto Driver Object is not full, the job will be listed into that queue in order of its priority. As soon as the Crypto Driver Object is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

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

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

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

[SWS_Csm_01041]⌈



](

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

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

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

7.2.2.3 Key Management

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

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

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

[SWS_Csm_01022] [***Crypto Service:***	***key element:***	***key element Name:***	***key element ID:***	***Mandatory:***
MAC	Key Material	`CRYPTO_KE_MAC_KEY`	1	x
	Proof (SHE)	`CRYPTO_KE_MAC_PROOF`	2	
Signature	Key Material	`CRYPTO_KE_SIGNATURE_KEY`	1	x
Random	Seed State	`CRYPTO_KE_RANDOM_SEED_STATE`	3	
	Algorithm	`CRYPTO_KE_RANDOM_ALGORITHM`	4	
Cipher/AEAD	Key Material	`CRYPTO_KE_CIPHER_KEY`	1	x
	Init Vector	`CRYPTO_KE_CIPHER_IV`	5	
	Proof (SHE)	`CRYPTO_KE_CIPHER_PROOF`	6	
	2nd Key Material	`CRYPTO_KE_CIPHER_2NDKEY`	7	

Key Exchange	Base	CRYPTO_KE_KEYEXCHANGE_BASE	8	x
	Private Key	CRYPTO_KE_KEYEXCHANGE_PRIVKEY	9	x
	Own Public Key	CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY	10	x
	Shared Value	CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE	1	x
	Algorithm	CRYPTO_KE_KEYEXCHANGE_ALGORITHM	12	
Key Derivation	Password	CRYPTO_KE_KEYDERIVATION_PASSWORD	1	x
	Salt	CRYPTO_KE_KEYDERIVATION_SALT	13	
	Iterations	CRYPTO_KE_KEYDERIVATION_ITERATIONS	14	
	Algorithm	CRYPTO_KE_KEYDERIVATION_ALGORITHM	15	
Key Generate	Key Material	CRYPTO_KE_KEYGENERATE_KEY	1	x
	Seed	CRYPTO_KE_KEYGENERATE_SEED	16	
	Algorithm	CRYPTO_KE_KEYGENERATE_ALGORITHM	17	
Certificate Parsing	Certificate	CRYPTO_KE_CERTIFICATE_DATA	0	x
	Format	CRYPTO_KE_CERTIFICATE_PARSING_FORMAT	18	
	Current Time	CRYPTO_KE_CERTIFICATE_CURRENT_TIME	19	
	Version	CRYPTO_KE_CERTIFICATE_VERSION	20	
	Serial Number	CRYPTO_KE_CERTIFICATE_SERIALNUMBER	21	
	Signature Algroithm	CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM	22	
	Issuer	CRYPTO_KE_CERTIFICATE_ISSUER	23	
	Validity start	CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE	24	
	Validity end	CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER	25	
	Subject	CRYPTO_KE_CERTIFICATE_SUBJECT	26	
	Subject Public Key	CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY	1	
	Extensions	CRYPTO_KE_CERTIFICATE_EXTENSIONS	27	
	Signature	CRYPTO_KE_CERTIFICATE_SIGNATURE	28	

]()

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

[SWS_Csm_00951] ⌈If the key material itself consist of more than one element, it shall be stored as PKCS#8 in the key element.

Examples are asymmetric algorithms like in RSA where the key consists of a modulus and an exponent or and an ECC key which consists of the X and Y coordinates.
⌋()

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

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

7.3 Error Classification

7.3.1 Development Errors

[SWS_Csm_00828]⌈Development Error Types

Type of error	*Related error code*	*Value [hex]*
API request called with invalid parameter (Nullpointer)	CSM_E_PARAM_POINTER	0x01
API request called before initialization of CSM module	CSM_E_UNINIT	0x05
Initialization of CSM module failed	CSM_E_INIT_FAILED	0x07
Requested service is not initialized	CSM_E_SERVICE_NOT_STARTED	0x09

⌋(SRS_CryptoStack_00086)

7.3.2 Runtime Errors

There are no runtime errors.

7.3.3 Transient Faults

There are no transient faults.

7.3.4 Production Errors

There are no production errors.

7.3.5 Extended Production Errors

There are no extended production errors.

7.4 Error detection

[SWS_Csm_00830] [If the API returns `CRYPTO_E_SMALL_BUFFER`, additonally `CSM_E_SMALL_BUFFER` shall be reported to the Det.
](SRS_CryptoStack_00087)

[SWS_Csm_00489] [The following table specifies which DET error values shall be reported for each API call:
](SRS_BSW_00406, SRS_BSW_00337, SRS_CryptoStack_00087, SRS_CryptoStack_00088)

[SWS_Csm_00539] [

API call	***Error condition***	***DET related error value***
All APIs except Csm_Init()	CSM is not initialized	CSM_E_UNINIT
Csm_Init	Initialization of CSM failed	CSM_E_INIT_FAILED
All APIs that have a pointer as parameter	Pointer is Nullpointer	CSM_E_PARAM_POINTER In case a NULL pointer has been passed, the API service shall report development error to DET if enabled and return immediately without any further action.
All APIs that have a keyId as parameter	keyId is out of range	`CSM_E_PARAM_HANDLE`

](SRS_BSW_00406, SRS_BSW_00337, SRS_BSW_00385, SRS_CryptoStack_00087, SRS_CryptoStack_00088)

8 API Specification

8.1 Imported types

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

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

Range:	`CRYPTO_E_BUSY`	`0x02`	The service request failed because the service is still busy
	`CRYPTO_E_SMALL_BUFFER`	`0x03`	The service request failed because the provided buffer is too small to store the result
	`CRYPTO_E_ENTROPY_EXHAUSTION`	`0x04`	The service request failed because the entropy of the random number generator is exhausted
	`CRYPTO_E_QUEUE_FULL`	`0x05`	The service request failed because the queue is full
	`CRYPTO_E_KEY_READ_FAIL`	`0x06`	The service request failed because read access was denied
	`CRYPTO_E_KEY_WRITE_FAIL`	`0x07`	The service request failed because the writing access failed
	`CRYPTO_E_KEY_NOT_AVAILABLE`	`0x08`	The service request failed because the key is not available
	`CRYPTO_E_KEY_NOT_VALID`	`0x09`	The service request failed because the key is invalid.
	`CRYPTO_E_KEY_SIZE_MISMATCH`	`0x0A`	The service request failed because the key size does not match.
	`CRYPTO_E_COUNTER_OVERFLOW`	`0x0B`	The service request failed because the counter is overflowed.
	`CRYPTO_E_JOB_CANCELED`	`0x0C`	The service request failed because the Job has been canceled.
Description:	Csm module specific return values for use in Std_ReturnType.		

⌋ (SRS_CryptoStack_00095)

8.2 Type Definitions

8.2.1 Crypto_AlgorithmFamilyType

[SWS_Csm_01047] ⌈

Name:	`Crypto_AlgorithmFamilyType`		
Type:	`Enumeration`		
Range:	`CRYPTO_ALGOFAM_NOT_SET`	`0x00`	Algorithm family is not set
	`CRYPTO_ALGOFAM_SHA1`	`0x01`	SHA1 hash
	`CRYPTO_ALGOFAM_SHA2_224`	`0x02`	SHA2-224 hash
	`CRYPTO_ALGOFAM_SHA2_256`	`0x03`	SHA2-256 hash
	`CRYPTO_ALGOFAM_SHA2_384`	`0x04`	SHA2-384 hash
	`CRYPTO_ALGOFAM_SHA2_512`	`0x05`	SHA2-512 hash
	`CRYPTO_ALGOFAM_SHA2_512_224`	`0x06`	SHA2-512/224 hash

	CRYPTO_ALGOFAM_SHA2_512_256	0x07	SHA2-512/256 hash
	CRYPTO_ALGOFAM_SHA3_224	0x08	SHA3-224 hash
	CRYPTO_ALGOFAM_SHA3_256	0x09	SHA3-256 hash
	CRYPTO_ALGOFAM_SHA3_384	0x0a	SHA3-384 hash
	CRYPTO_ALGOFAM_SHA3_512	0x0b	SHA3-512 hash
	CRYPTO_ALGOFAM_SHAKE128	0x0c	SHAKE128 hash
	CRYPTO_ALGOFAM_SHAKE256	0x0d	SHAKE256 hash
	CRYPTO_ALGOFAM_RIPEMD160	0x0e	RIPEMD hash
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0f	BLAKE-1-256 hash
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	BLAKE-1-512 hash
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	BLAKE-2s-256 hash
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	BLAKE-2s-512 hash
	CRYPTO_ALGOFAM_3DES	0x13	3DES cipher
	CRYPTO_ALGOFAM_AES	0x14	AES cipher
	CRYPTO_ALGOFAM_CHACHA	0x15	ChaCha cipher
	CRYPTO_ALGOFAM_RSA	0x16	RSA cipher
	CRYPTO_ALGOFAM_ED25519	0x17	ED22518 elliptic curve
	CRYPTO_ALGOFAM_BRAINPOOL	0x18	Brainpool elliptic curve
	CRYPTO_ALGOFAM_ECCNIST	0x19	NIST ECC elliptic curves
	CRYPTO_ALGOFAM_SECURECOUNTER	0x1a	Secure Counter
	CRYPTO_ALGOFAM_RNG	0x1b	Random Number Generator
	CRYPTO_ALGOFAM_SIPHASH	0x1c	SipHash
	CRYPTO_ALGOFAM_ECIES	0x1d	ECIES Cipher
	CRYPTO_ALGOFAM_CUSTOM	0xff	Custom algorithm family
Description:	Enumeration of the algorithm family.		

] ()

8.3 Function Definitions

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

8.3.1 General Interface

8.3.1.1 Csm_Init

[SWS_Csm_00646] [

Service name:	Csm_Init

Syntax:	`void Csm_Init(`
	`  void`
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Initializes the CSM module.

] (SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)

[SWS_Csm_00659] [If the initialization of the CSM module fails, the CSM shall report CSM_E_INIT_FAILED to the DET.
]()

8.3.1.2 Csm_GetVersionInfo

[SWS_Csm_00705] [

Service name:	Csm_GetVersionInfo	

Syntax:	`void Csm_GetVersionInfo(`
	`  Std_VersionInfoType* versioninfo`
	`)`	

Service ID[hex]:	0x3b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

] (SRS_BSW_00407)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_GetVersionInfo`.

8.3.2 Hash Interface

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

8.3.2.1 Csm_Hash

[SWS_Csm_00980] [

Service name:	Csm_Hash	

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

Service ID[hex]:	0x5d	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.

	mode	Indicates which operation mode(s) to perfom.
	dataPtr	Contains the pointer to the data for which the hash shall be computed.
	dataLength	Contains the number of bytes to be hashed.
Parameters (inout):	resultLengthPtr	Holds a pointer to the memory location in which the output length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	resultPtr	Contains the pointer to the data where the hash value shall be stored.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
Description:	Uses the given data to perform the hash calculation and stores the hash.	

] (SRS_CryptoStack_00024)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_Hash`.

[SWS_Csm_01015] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.outputPtr = resultPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
resultLengthPtr.
```

] ( )

8.3.2.2 Csm_HashStart

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00089] [

Service name:	Csm_HashStart (obsolete)	
Syntax:	`Std_ReturnType Csm_HashStart( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x03	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration that has to be used during the hash value computation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated and does nothing. **Tags:**	

	atp.Status=obsolete

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_HashStart`.

8.3.2.3 Csm_HashUpdate

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00094] [

Service name:	Csm_HashUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_HashUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	Holds a pointer to the data to be hashed
	dataLength	Contains the number of bytes to be hashed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the hash service with the input data **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_HashUpdate`.

8.3.2.4 Csm_HashFinish

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00101] [

Service name:	Csm_HashFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_HashFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr,`
	`  boolean TruncationIsAllowed`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	TruncationIsAllowed	This parameter states whether a truncation of the result is

		allowed or not. TRUE: truncation is allowed. FALSE: truncation is not allowed.
Parameters (inout):	resultLengthPtr	holds a pointer to the memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	resultPtr	holds a pointer to the memory location which will hold the result of the hash value computation. If the result does not fit into the given buffer, and truncation is allowed, the result shall be truncated.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result, and truncation was not allowed.
Description:	This function is deprecated. Finishes the hash service of the CSM module and stores the hash. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_HashFinish`.

8.3.2.5 Checksum Interface

DEPRECATED: These functions will be removed in the next major release! It shall use the hash service to calculate the checksum.

8.3.2.5.1 Csm_ChecksumStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00335] [

Service name:	Csm_ChecksumStart (obsolete)	

Syntax:	`Std_ReturnType Csm_ChecksumStart(`
	`  Csm_ConfigIdType cfgId`
	`)`	

Service ID[hex]:	0x28	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the checksum computation
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated and does nothing. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_ChecksumStart`.

8.3.2.5.2 Csm_ChecksumUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00341] [

Service name:	Csm_ChecksumUpdate (obsolete)	
Syntax:	`Std_ReturnType Csm_ChecksumUpdate( Csm_ConfigIdType cfgId, const uint8* dataPtr, uint32 dataLength )`	
Service ID[hex]:	0x29	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	holds a pointer to the data for which the checksum shall be calculated
	dataLength	contains the length of the input data in bytes
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the checksum service with the input data. It shall use the hash service to calculate the checksum. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_ChecksumUpdate`.

8.3.2.5.3 Csm_ChecksumFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00348] [

Service name:	Csm_ChecksumFinish (obsolete)	
Syntax:	`Std_ReturnType Csm_ChecksumFinish( Csm_ConfigIdType cfgId, uint8* resultPtr, uint32* resultLengthPtr, boolean TruncationIsAllowed )`	
Service ID[hex]:	0x2a	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	TruncationIsAllowed	This parameter states whether a truncation of the result is allowed or not. TRUE: truncation is allowed.

		FALSE: truncation is not allowed.
Parameters (inout):	resultLengthPtr	holds a pointer to the memory location in which the length information 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 computed checksum shall be stored
Parameters (out):	resultPtr	holds a pointer to the memory location which will hold the result of the checksum calculation. If the result does not fit into the given buffer, and truncation is allowed, the result shall be truncated
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result, and truncation was not allowed.
Description:	This function is deprecated. Finishes the checksum service of the CSM module and stores the hash. It shall use the hash service to calculate the checksum. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_ChecksumFinish`.

8.3.3 MAC interface

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

8.3.3.1 Csm_MacGenerate

[SWS_Csm_00982] [

Service name:	Csm_MacGenerate	

Syntax:	`Std_ReturnType Csm_MacGenerate(`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  uint8* macPtr,`
	`  uint32* macLengthPtr`
	`)`	

Service ID[hex]:	0x60	
Sync/Async:	Asynchronous or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perfom.
	dataPtr	Contains the pointer to the data for which the MAC shall be computed.

	dataLength	Contains the number of bytes to be hashed.
Parameters (inout):	macLengthPtr	Holds a pointer to the memory location in which the output length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer provided by macPtr. When the request has finished, the actual length of the returned MAC shall be stored.
Parameters (out):	macPtr	Contains the pointer to the data where the MAC shall be stored.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
Description:	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	

] (SRS_CryptoStack_00022)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacGenerate`.

[SWS_Csm_01017] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.outputPtr = resultPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
resultLengthPtr,
```

] ( )

8.3.3.2 Csm_MacGenerateStart

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00108] [

Service name:	Csm_MacGenerateStart (obsolete)	
Syntax:	`Std_ReturnType Csm_MacGenerateStart( Csm_ConfigIdType cfgId, const Csm_SymKeyType* keyPtr )`	
Service ID[hex]:	0x06	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration which has to be used during the MAC computation.
	keyPtr	Holds a pointer to the key necessary for the MAC generation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for MAC generation.	

	Tags: atp.Status=obsolete

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacGenerateStart`.

8.3.3.3 Csm_MacGenerateUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00114] [

Service name:	Csm_MacGenerateUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_MacGenerateUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	holds a pointer to the data for which a MAC shall be computed.
	dataLength	contains the number of bytes for which the MAC shall be computed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the MAC generate service with the input data. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacGenerateUpdate`.

8.3.3.4 Csm_MacGenerateFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00121] [

Service name:	Csm_MacGenerateFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_MacGenerateFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr,`
	`  boolean TruncationIsAllowed`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has

		to be used during the operation.
	TruncationIsAllowed	This parameter states whether a truncation of the result is allowed or not. TRUE: truncation is allowed. FALSE: truncation is not allowed.
Parameters (inout):	resultLengthPtr	holds a pointer to the memory location in which the length information 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 MAC shall be stored.
Parameters (out):	resultPtr	holds a pointer to the memory location which will hold the result of the MAC generation. If the result does not fit into the given buffer, and truncation is allowed, the result shall be truncated
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result, and truncation was not allowed.
Description:	This function is deprecated. Finishes the MAC generation service and stores the MAC in the memory location pointed by the MAC pointer. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacGenerateFinish`.

8.3.3.5 Csm_MacVerify

[SWS_Csm_01050] [

Service name:	Csm_MacVerify	

Syntax:	`Std_ReturnType Csm_MacVerify(`
	`  uint32 jobId,`
	`  Crypto_OperationModeType mode,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength,`
	`  const uint8* macPtr,`
	`  const uint32 macLength,`
	`  Crypto_VerifyResultType* verifyPtr`
	`)`	

Service ID[hex]:	0x61	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Indicates which operation mode(s) to perfom.
	mode	Indicates which operation mode(s) to perfom.
	dataPtr	Holds a pointer to the data for which the MAC shall be verified.
	dataLength	Contains the number of data bytes for which the MAC shall be verified.
	macPtr	Holds a pointer to the MAC to be verified.
	macLength	Contains the MAC length in BITS to be verified.
Parameters (inout):	None	
Parameters (out):	verifyPtr	Holds a pointer to the memory location, which will hold the result of the MAC verification.

Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid"
Description:	Verifies the given MAC by comparing if the MAC is generated with the given data.	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacVerify`.

[SWS_Csm_01016] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.secondaryInputPtr = macPtr,
job->jobPrimitiveInputOutput.secondaryInputLength = macLength,
job->jobPrimitiveInputOutput.verifyPtr = verifyPtr.
```

] ( )

8.3.3.6 Csm_MacVerifyStart

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00128] [

Service name:	Csm_MacVerifyStart (obsolete)	

Syntax:	`Std_ReturnType Csm_MacVerifyStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_SymKeyType* keyPtr`
	`)`	

Service ID[hex]:	0x09	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the MAC verification.
	keyPtr	holds a pointer to the key necessary for the MAC verification.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for MAC verification. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacVerifyStart`.

8.3.3.7 Csm_MacVerifyUpdate

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00134] [

Service name:	Csm_MacVerifyUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_MacVerifyUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength`
	`)`	

Service ID[hex]:	0x0a	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	holds a pointer to the data for which a MAC shall be verified.
	dataLength	contains the number of bytes for which the MAC shall be verified.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the MAC verification service with the input data. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacVerifyUpdate`.

8.3.3.8 Csm_MacVerifyFinish

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00141] [

Service name:	Csm_MacVerifyFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_MacVerifyFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* MacPtr,`
	`  uint32 MacLength,`
	`  Csm_VerifyResultType* resultPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	MacPtr	holds a pointer to the memory location which will hold the MAC to verify.
	MacLength	holds the length of the MAC to be verified. Note: the computed MAC will be internally truncated to this length.
Parameters (inout):	None	
Parameters (out):	resultPtr	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

		CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Finishes the MAC verification and stores the verification result in the memory location pointed by the result pointer. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_MacVerifyFinish`.

8.3.4 Cipher Interface

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

8.3.4.1 Csm_Encrypt

[SWS_Csm_00984] [

Service name:	Csm_Encrypt	

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

Service ID[hex]:	0x5e	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perfom.
	dataPtr	Contains the pointer to the data to be encrypted.
	dataLength	Contains the number of bytes to encrypt.
Parameters (inout):	resultLengthPtr	Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	resultPtr	Contains the pointer to the data where the encrypted data shall be stored.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
Description:	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.	

] (SRS_CryptoStack_00020, SRS_CryptoStack_00021)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_Encrypt`.

[SWS_Csm_00986] ⌈The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:
```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.outputPtr = resultPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
resultLengthPtr.
```
⌋ ( )

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

8.3.4.2 Csm_Decrypt

[SWS_Csm_00989] ⌈

Service name:	Csm_Decrypt	
Syntax:	`Std_ReturnType Csm_Decrypt( uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr )`	
Service ID[hex]:	0x5f	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perfom.
	dataPtr	Contains the pointer to the data to be decrypted.
	dataLength	Contains the number of bytes to decrypt.
Parameters (inout):	resultLengthPtr	Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	resultPtr	Contains the pointer to the memory location where the decrypted data shall be stored.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
Description:	Decrypts the given encrypted data and store the decrypted plaintext in the memory location pointed by the result pointer.	

⌋ (SRS_CryptoStack_00020, SRS_CryptoStack_00021)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_Decrypt`.

[SWS_Csm_00990] ⌈The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.outputPtr = resultPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
resultLengthPtr.
```

⌋ ( )

8.3.4.3 Symmetrical block interface

A block cipher is a symmetric key cipher operating on fixed-length blocks, with an unvarying transformation. A block cipher encryption algorithm might take (for example) a 128-bit block of plaintext as input, and output a corresponding 128-bit block of ciphertext. The exact transformation is controlled using a second input — the secret key. Decryption is similar: the decryption algorithm takes, in this example, a 128-bit block of ciphertext together with the secret key, and yields the original 128-bit block of plaintext.

8.3.4.3.1 Csm_SymBlockEncryptStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00168] ⌈

Service name:	Csm_SymBlockEncryptStart (obsolete)	
Syntax:	`Std_ReturnType Csm_SymBlockEncryptStart( Csm_ConfigIdType cfgId, const Csm_SymKeyType* keyPtr )`	
Service ID[hex]:	0x10	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the symmetrical block encryption computation.
	keyPtr	holds a pointer to the key which has to be used during the symmetrical block encryption computation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key and initialization vector for symmetrical encryption. **Tags:** atp.Status=obsolete	

⌋ ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymBlockEncryptStart`.

8.3.4.3.2 Csm_SymBlockEncryptUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00173] ⌈

Service name:	Csm_SymBlockEncryptUpdate	
Syntax:	`Std_ReturnType Csm_SymBlockEncryptUpdate( Csm_ConfigIdType cfgId, const uint8* plainTextPtr, uint32 plainTextLength, uint8* cipherTextPtr, uint32* cipherTextLengthPtr )`	
Service ID[hex]:	0x11	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	plainTextPtr	holds a pointer to the plain text that shall be encrypted.
	plainTextLength	contains the length of the plain text in bytes
Parameters (inout):	cipherTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by cipherTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out):	cipherTextPtr	holds a pointer to the memory location which will hold the encrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Feeds the symmetrical encrypt service with the input data and store the ciphertext in the memory location pointed by the ciphertext pointer.	

⌋ ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymBlockEncryptUpdate`.

8.3.4.3.3 Csm_SymBlockEncryptFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00180] ⌈

Service name:	Csm_SymBlockEncryptFinish	
Syntax:	`Std_ReturnType Csm_SymBlockEncryptFinish( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x12	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Finishes the symmetrical encrypt service.	

] () Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymBlockEncryptFinish`.

8.3.4.3.4 Csm_SymBlockDecryptStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00187] [

Service name:	Csm_SymBlockDecryptStart (obsolete)	

Syntax:	`Std_ReturnType Csm_SymBlockDecryptStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_SymKeyType* keyPtr`
	`)`	

Service ID[hex]:	0x13	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the constant CSM module configuration which has to be used during the symmetrical block decryption computation
	keyPtr	holds a pointer to the key which has to be used during the symmetrical block decryption computation
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for symmetrical block decryption. **Tags:** atp.Status=obsolete	

] () Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymBlockDecryptStart`.

8.3.4.3.5 Csm_SymBlockDecryptUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00192] [

Service name:	Csm_SymBlockDecryptUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_SymBlockDecryptUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* cipherTextPtr,`
	`  uint32 cipherTextLength,`
	`  uint8* plainTextPtr,`
	`  uint32* plainTextLengthPtr`
	`)`	

Service ID[hex]:	0x14	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.

	cipherTextPtr	holds a pointer to the constant cipher text that shall be decrypted.
	cipherTextLength	contains the length of the cipher text in bytes.
Parameters (inout):	plainTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out):	plainTextPtr	holds a pointer to the memory location which will hold the decrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Feeds the symmetrical block decrypt service with the input data and store the decrypted plaintext in the memory location pointed by the plaintext pointer. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00700] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymBlockDecryptUpdate`.

8.3.4.3.6 Csm_SymBlockDecryptFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00199] [

Service name:	Csm_SymBlockDecryptFinish (obsolete)	
Syntax:	`Std_ReturnType Csm_SymBlockDecryptFinish( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x15	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Finishes the symmetrical block decrypt service **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymBlockDecryptFinish`.

8.3.4.4 Symmetrical interface

Symmetric-key algorithms are algorithms that use identical cryptographic keys for both decryption and encryption. The keys, in practice, represent a shared secret between two or more parties.

8.3.4.4.1 Csm_SymEncryptStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00206] ⌈

Service name:	Csm_SymEncryptStart (obsolete)	

Syntax:	`Std_ReturnType Csm_SymEncryptStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_SymKeyType* keyPtr,`
	`  const uint8* InitVectorPtr,`
	`  uint32 InitVectorLength`
	`)`	

Service ID[hex]:	0x16	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the symmetrical encryption computation.
	keyPtr	holds a pointer to the key which has to be used during the symmetrical encryption computation
	InitVectorPtr	holds a pointer to the initialisation vector which has to be used during the symmetrical encryption computation
	InitVectorLength	holds the length of the initialisation vector which has to be used during the symmetrical encryption computation
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This interface shall be used to initialize the symmetrical encrypt service of the CSM module. If the service state is "active", the function shall return with "CSM_E_BUSY". Otherwise, this function shall store the given configuration information which is identified by "cfgId", call the function Cry_<Primitive>Start of the primitive which is identified by the "cfgId" and return the value returned by that function. If Cry_<Primitive>Start returned successfully, the service state has to be set to "active". **Tags:** atp.Status=obsolete	

⌋ ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymEncryptStart`.

8.3.4.4.2 Csm_SymEncryptUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00212] [

Service name:	Csm_SymEncryptUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_SymEncryptUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* plainTextPtr,`
	`  uint32 plainTextLength,`
	`  uint8* cipherTextPtr,`
	`  uint32* cipherTextLengthPtr`
	`)`	

Service ID[hex]:	0x17	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	plainTextPtr	holds a pointer to the plain text that shall be encrypted.
	plainTextLength	contains the length of the plain text in bytes
Parameters (inout):	cipherTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by cipherTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out):	cipherTextPtr	holds a pointer to the memory location which will hold the encrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This interface shall be used to feed the symmetrical encryption service with the input data. If the service state is "idle", the function has to return with "E_NOT_OK". Otherwise, this function shall call the function Cry_<Primitive>Update of the primitive which is identified by the stored configuration information and return the value returned by that function. The encryption process is done by the underlying primitive. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00665] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymEncryptUpdate`.

8.3.4.4.3 Csm_SymEncryptFinish

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00221] ⌈

Service name:	Csm_SymEncryptFinish	

Syntax:	`Std_ReturnType Csm_SymEncryptFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* cipherTextPtr,`
	`  uint32* cipherTextLengthPtr`
	`)`	

Service ID[hex]:	0x18	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	cipherTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by cipherTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out):	cipherTextPtr	holds a pointer to the memory location which will hold the encrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This interface shall be used to finish the symmetrical encryption service. If the service state is "idle", the function has to return with "E_NOT_OK". Otherwise, this function shall call the function Cry_<Primitive>Finish of the primitive which is identified by the stored configuration information and return the value returned by that function. The encryption process is done by the underlying primitive.	

⌋ ()

[SWS_Csm_00666] ⌈The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
⌋()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymEncryptFinish`.

8.3.4.4.4 Csm_SymDecryptStart

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00228] ⌈

Service name:	Csm_SymDecryptStart (obsolete)

Syntax:	`Std_ReturnType Csm_SymDecryptStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_SymKeyType* keyPtr,`
	`  const uint8* InitVectorPtr,`
	`  uint32 InitVectorLength`
	`)`

Service ID[hex]:	0x19
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)
Reentrancy:	Non Reentrant

Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the symmetrical decryption computation
	keyPtr	holds a pointer to the key which has to be used during the symmetrical decryption computation
	InitVectorPtr	holds a pointer to the initialisation vector which has to be used during the symmetrical decryption computation
	InitVectorLength	holds the length of the initialisation vector which has to be used during the symmetrical decryption computation
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key and initialization vector for symmetrical decryption. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymDecryptStart`.

8.3.4.4.5 Csm_SymDecryptUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00234] [

Service name:	Csm_SymDecryptUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_SymDecryptUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* cipherTextPtr,`
	`  uint32 cipherTextLength,`
	`  uint8* plainTextPtr,`
	`  uint32* plainTextLengthPtr`
	`)`	

Service ID[hex]:	0x1a	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	cipherTextPtr	holds a pointer to the cipher text that shall be decrypted.
	cipherTextLength	contains the length of the cipher text in bytes.
Parameters (inout):	plainTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out):	plainTextPtr	holds a pointer to the memory location which will hold the decrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Feeds the symmetrical decrypt service with the input	

	data and store the decrypted plaintext in the memory location pointed by the plaintext pointer. **Tags:** atp.Status=obsolete

] ()

[SWS_Csm_00667] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymDecryptUpdate`.

8.3.4.4.6 Csm_SymDecryptFinish

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00243] [

Service name:	Csm_SymDecryptFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_SymDecryptFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* plainTextPtr,`
	`  uint32* plainTextLengthPtr`
	`)`	

Service ID[hex]:	0x1b	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	plainTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out):	plainTextPtr	holds a pointer to the memory location which will hold the decrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Finishes the symmetrical decrypt service. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00668] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()
Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymDecryptFinish`.

8.3.4.5 Asymmetrical interface

Asymmetric-key algorithms are algorithms that use pairs of cryptographic keys (public and private keys) for decryption and encryption. The private key, in practice, represent a secret while the public key can be made publically available.

8.3.4.5.1 Csm_AsymEncryptStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00250] [

Service name:	Csm_AsymEncryptStart (obsolete)	
Syntax:	`Std_ReturnType Csm_AsymEncryptStart( Csm_ConfigIdType cfgId, const Csm_AsymPublicKeyType* keyPtr )`	
Service ID[hex]:	0x1c	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the asymmetrical encryption computation
	keyPtr	holds a pointer to the key necessary for the asymmetrical computation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for asymmetrical encryption. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymEncryptStart`.

8.3.4.5.2 Csm_AsymEncryptUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00256] [

Service name:	Csm_AsymEncryptUpdate (obsolete)	
Syntax:	`Std_ReturnType Csm_AsymEncryptUpdate( Csm_ConfigIdType cfgId, const uint8* plainTextPtr, uint32 plainTextLength, uint8* cipherTextPtr, uint32* cipherTextLengthPtr )`	
Service ID[hex]:	0x1d	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	plainTextPtr	holds a pointer to the plain text that shall be encrypted.
	plainTextLength	contains the length of the plain text in bytes

Parameters (inout):	cipherTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by cipherTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out):	cipherTextPtr	holds a pointer to the memory location which will hold the encrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Feeds the asymmetrical encrypt service with the input data and store the ciphertext in the memory location pointed by the ciphertext pointer. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00669] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymEncryptUpdate`.

8.3.4.5.3 Csm_AsymEncryptFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00265] [

Service name:	Csm_AsymEncryptFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_AsymEncryptFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* cipherTextPtr,`
	`  uint32* cipherTextLengthPtr`
	`)`	

Service ID[hex]:	0x1e	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	cipherTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by cipherTextPtr. When the request has finished, the amount of data that has been encrypted shall be stored.
Parameters (out):	cipherTextPtr	holds a pointer to the memory location which will hold the encrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Finishes the asymmetrical encrypt service and store	

	the ciphertext in the memory location pointed by the ciphertext pointer. **Tags:** atp.Status=obsolete

] ()

[SWS_Csm_00670] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()

8.3.4.5.4 Csm_AsymDecryptStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00272] [

Service name:	Csm_AsymDecryptStart (obsolete)	

Syntax:	`Std_ReturnType Csm_AsymDecryptStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_AsymPrivateKeyType* keyPtr`
	`)`	

Service ID[hex]:	0x1f	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the asymmetrical decryption computation
	keyPtr	holds a pointer to the key necessary for the asymmetrical computation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for asymmetrical decryption. It also sets the given key with Csm_KeyElementSet(). **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function Csm_AsymDecryptStart.

8.3.4.5.5 Csm_AsymDecryptUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00278] [

Service name:	Csm_AsymDecryptUpdate (obsolete)

Syntax:	`Std_ReturnType Csm_AsymDecryptUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* cipherTextPtr,`
	`  uint32 cipherTextLength,`
	`  uint8* plainTextPtr,`
	`  uint32* plainTextLengthPtr`
	`)`

Service ID[hex]:	0x20

Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	cipherTextPtr	holds a pointer to the encrypted data
	cipherTextLength	contains the length of the encrypted data in bytes.
Parameters (inout):	plainTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out):	plainTextPtr	holds a pointer to the memory location which will hold the decrypted text.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Feeds the asymmetrical decrypt service with the input data and stores the encrypted plaintext in the memory location pointed by the plaintext pointer. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00671] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
]()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymDecryptUpdate`.

8.3.4.5.6 Csm_AsymDecryptFinish

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00287] [

Service name:	Csm_AsymDecryptFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_AsymDecryptFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* plainTextPtr,`
	`  uint32* plainTextLengthPtr`
	`)`	

Service ID[hex]:	0x21	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	plainTextLengthPtr	holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by plainTextPtr. When the request has finished, the amount of data that has been decrypted shall be stored.
Parameters (out):	plainTextPtr	holds a pointer to the memory location which will hold the decrypted text.

Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Finshes the asymmetrical decrypt service amd store the decrypted plaintext in the memory location pointed by the plaintext pointer. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00672] ⌈The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small, `CRYPTO_E_SMALL_BUFFER` shall be returned.
⌋()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymDecryptFinish`.

8.3.5 Authenticated Encryption with Associated Data (AEAD) Interface

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

8.3.5.1 Csm_AEADEncrypt

[SWS_Csm_01023] ⌈

Service name:	Csm_AEADEncrypt	

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

Service ID[hex]:	0x62	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perfom.
	plaintextPtr	Contains the pointer to the data to be encrypted.
	plaintextLength	Holds a pointer to the memory location in which the output length in bytes of the paintext is stored. On calling this function, this parameter shall contain the size of the buffer provided by plaintextPtr. When the request has finished, the actual length of the returned value shall be stored.
	associatedDataPtr	Contains the pointer to the associated data.

	associatedDataLengthPtr	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_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid"
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.	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AEADEncrypt`.

[SWS_Csm_01025] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = plaintextPtr,
job->jobPrimitiveInputOutput.inputLength = plaintextLength,
job->jobPrimitiveInputOutput.secondaryInputPtr =
associatedDataPtr,
job->jobPrimitiveInputOutput.secondaryInputLength =
associatedDataLengthPtr,
job->jobPrimitiveInputOutput.outputPtr = ciphertextPtr,
job->jobPrimitiveInputOutput.outputLength = ciphertextLength,
job->jobPrimitiveInputOutput.secondaryOutputPtr = tagPtr,
job->jobPrimitiveInputOutput.secondaryOutputLengthPtr =
tagLengthPtr.
```

] ( )

8.3.5.2 Csm_AEADDecrypt

[SWS_Csm_01026] [

Service name:	Csm_AEADDecrypt
Syntax:	`Std_ReturnType Csm_AEADDecrypt(` `  uint32 jobId,`

	`Crypto_OperationModeType mode,`
	`const uint8* ciphertextPtr,`
	`uint32 ciphertextLength,`
	`const uint8* associatedDataPtr,`
	`uint32 associtatedDataLength,`
	`const uint8* tagPtr,`
	`uint32 tagLength,`
	`uint8* plaintextPtr,`
	`uint32* plaintextLengthPtr,`
	`Crypto_VerifyResultType* verifyPtr`
	)	

Service ID[hex]:	0x63	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	Indicates which operation mode(s) to perfom.
	ciphertextPtr	Contains the pointer to the data to be decrypted.
	ciphertextLength	Contains the number of bytes to decrypt.
	associatedDataPtr	Contains the pointer to the associated data.
	associtatedDataLength	Contains the length in bytes of the associated data.
	tagPtr	Contains the pointer to the Tag to be verified.
	tagLength	Contains the length in bytes of the Tag to be verified.
Parameters (inout):	plaintextLengthPtr	Holds a pointer to the memory location in which the output length in bytes of the paintext is stored. On calling this function, this parameter shall contain the size of the buffer provided by plaintextPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	plaintextPtr	Contains the pointer to the data where the decrypted data shall be stored.
	verifyPtr	Contains the pointer to the result of the verification.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid"
Description:	Uses the given data to perform an AEAD encryption and stores the ciphertext and the MAC in the memory locations pointed by the ciphertext pointer and Tag pointer.	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AEADDecrypt`.

[SWS_Csm_01027] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = ciphertextPtr,
job->jobPrimitiveInputOutput.inputLength = ciphertextLength,
job->jobPrimitiveInputOutput.secondaryInputPtr =
associatedDataPtr,
job->jobPrimitiveInputOutput.secondaryInputLength =
associatedLength,
```

```
job->jobPrimitiveInputOutput.tertiaryInputPtr = tagPtr,
job->jobPrimitiveInputOutput.tertiaryInputLength = tagLength,
job->jobPrimitiveInputOutput.outputPtr = plaintextPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
plaintextLengthPtr.
```
] ( )

8.3.6 Signature Interface

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

8.3.6.1 Csm_SignatureGenerate

[SWS_Csm_00992] [

Service name:	Csm_SignatureGenerate	
Syntax:	`Std_ReturnType Csm_SignatureGenerate( uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, uint8* resultPtr, uint32* resultLengthPtr )`	
Service ID[hex]:	0x76	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	The Crypto_JobInfoType job with the corresponding jobId shall be modified in the following way:
	dataPtr	Contains the pointer to the data to be signed.
	dataLength	Contains the number of bytes to sign.
Parameters (inout):	resultLengthPtr	Contains the number of bytes of the associated data.
Parameters (out):	resultPtr	Contains the pointer to the data where the signature shall be stored.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
Description:	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.	

] (SRS_CryptoStack_00023)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureGenerate`.

[SWS_Csm_00993] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.outputPtr = resultPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
resultLengthPtr,
```

] ( )

8.3.6.2 Csm_SignatureGenerateStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00294] [

Service name:	Csm_SignatureGenerateStart (obsolete)	
Syntax:	`Std_ReturnType Csm_SignatureGenerateStart( Csm_ConfigIdType cfgId, const Csm_AsymPrivateKeyType* keyPtr )`	
Service ID[hex]:	0x22	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the signature generation
	keyPtr	holds a pointer to the key necessary for the signature generation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for signature generation. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureGenerateStart`.

8.3.6.3 Csm_SignatureGenerateUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00300] [

Service name:	Csm_SignatureGenerateUpdate (obsolete)
Syntax:	`Std_ReturnType Csm_SignatureGenerateUpdate( Csm_ConfigIdType cfgId, const uint8* dataPtr, uint32 dataLength )`
Service ID[hex]:	0x23
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)

Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	holds a pointer to the data that shall be signed
	dataLength	contains the length of the data to be signed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the signature generate service with the input data. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureGenerateUpdate`.

8.3.6.4 Csm_SignatureGenerateFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00307] [

Service name:	Csm_SignatureGenerateFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_SignatureGenerateFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* resultPtr,`
	`  uint32* resultLengthPtr`
	`)`	

Service ID[hex]:	0x24	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	resultLengthPtr	holds a pointer to the memory location in which the length information 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 computed signature shall be stored
Parameters (out):	resultPtr	holds a pointer to the memory location which will hold the result of the signature generation.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_SMALL_BUFFER: the provided buffer is too small to store the result
Description:	This function is deprecated. Finishes the signature generation service and store the signature in the memory location pointed by the result pointer. **Tags:** atp.Status=obsolete	

] ()

[SWS_Csm_00673] [The CSM shall check if the provided buffer is large enough to hold the result of the computation. If the provided buffer is too small,

`CRYPTO_E_SMALL_BUFFER` shall be returned.
](

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureGenerateFinish`.

8.3.6.5 Csm_SignatureVerify

[SWS_Csm_00996] [

Service name:	Csm_SignatureVerify	
Syntax:	`Std_ReturnType Csm_SignatureVerify( uint32 jobId, Crypto_OperationModeType mode, const uint8* dataPtr, uint32 dataLength, const uint8* singaturePtr, uint32 signatureLength, Crypto_VerifyResultType* verifyPtr )`	
Service ID[hex]:	0x64	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	mode	The Crypto_JobInfoType job with the corresponding jobId shall be modified in the following way:
	dataPtr	Contains the pointer to the data to be verified.
	dataLength	Contains the number of data bytes.
	singaturePtr	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_QUEUE_FULL: request failed, the queue is full CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
Description:	Verifies the given MAC by comparing if the signature is generated with the given data.	

] (SRS_CryptoStack_00023)

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureVerify`.

[SWS_Csm_00997] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be set in the following way:

```
job->jobPrimitiveInputOutput.mode = mode,
job->jobPrimitiveInputOutput.inputPtr = dataPtr,
job->jobPrimitiveInputOutput.inputLength = dataLength,
job->jobPrimitiveInputOutput.secondaryInputPtr = signaturePtr,
```

```
job->jobPrimitiveInputOutput.secondaryInputLength =
signatureLength,
job->jobPrimitiveInputOutput.verifyPtr = verifyPtr.
] ( )
```

8.3.6.6 Csm_SignatureVerifyStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00314] [

Service name:	Csm_SignatureVerifyStart (obsolete)	
Syntax:	`Std_ReturnType Csm_SignatureVerifyStart( Csm_ConfigIdType cfgId, const Csm_AsymPublicKeyType* keyPtr )`	
Service ID[hex]:	0x25	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the signature computation/verification
	keyPtr	holds a pointer to the key necessary for the signature verification.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Sets the key for signature verification. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureVerifyStart`.

8.3.6.7 Csm_SignatureVerifyUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00320] [

Service name:	Csm_SignatureVerifyUpdate (obsolete)	
Syntax:	`Std_ReturnType Csm_SignatureVerifyUpdate( Csm_ConfigIdType cfgId, const uint8* dataPtr, uint32 dataLength )`	
Service ID[hex]:	0x26	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	holds a pointer to the signature which shall be verified
	dataLength	contains the length of the signature to verify in bytes
Parameters (inout):	None	
Parameters (out):	None	

Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the signature verification service with the input data. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureVerifyUpdate`.

8.3.6.8 Csm_SignatureVerifyFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00327] [

Service name:	Csm_SignatureVerifyFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_SignatureVerifyFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* signaturePtr,`
	`  uint32 signatureLength,`
	`  Csm_VerifyResultType* resultPtr`
	`)`	

Service ID[hex]:	0x27	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	signaturePtr	holds a pointer to the memory location which holds the signature to be verified
	signatureLength	holds the length of the Signature to be verified
Parameters (inout):	None	
Parameters (out):	resultPtr	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 CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Finishes the signature verification service and stores the signature in the memory location pointed by the result pointer. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SignatureVerifyFinish`.

8.3.7 Secure Counter Interface

A secure counter is a counter which should be secured in a special way, e.g. inside an HSM. It is only possible to increment and read the counter.

8.3.7.1 Csm_SecureCounterIncrement

[SWS_Csm_00998] ⌈

Service name:	Csm_SecureCounterIncrement	
Syntax:	`Std_ReturnType Csm_SecureCounterIncrement( uint32 jobId, uint64 stepSize )`	
Service ID[hex]:	0x65	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
	stepSize	Holds the value by which the counter will be incremented.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, job is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_COUNTER_OVERFLOW: the counter is overflowed
Description:	Increments the value of the secure counter by the value contained in stepSize.	

⌋ (SRS_CrytptoStack_00030)

[SWS_Csm_00973] ⌈⌈ If no errors are detected by Csm, the service `Csm_SecureCounterIncrement()` shall call `CryIf_SecureCounterIncrement()`.
⌋()

8.3.7.2 Csm_SecureCounterRead

[SWS_Csm_00999] ⌈

Service name:	Csm_SecureCounterRead	
Syntax:	`Std_ReturnType Csm_SecureCounterRead( uint32 jobId, uint64* counterValuePtr )`	
Service ID[hex]:	0x66	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
Parameters (inout):	None	
Parameters (out):	counterValuePtr	Holds a pointer to the memory location which shall hold the value of the secure counter
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, job is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full
Description:	Retrieves the value of a secure counter.	

⌋ (SRS_CrytptoStack_00030)

[SWS_Csm_01000] ⌈ If no errors are detected by Csm, the service `Csm_SecureCounterRead()` shall call `CryIf_SecureCounterRead()`. The `Crypto_JobInfoType` job with the corresponding `jobId` shall be used as

parameter in `CryIf_SecureCounterRead()` and shall be filled in the following way:
`job->jobPrimitiveInputOutput.output64Ptr = counterValuePtr`
] ( )

8.3.8 Random Interface

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

8.3.8.1 Csm_RandomGenerate

[SWS_Csm_01543] [

Service name:	Csm_RandomGenerate	
Syntax:	`Std_ReturnType Csm_RandomGenerate( uint32 jobId, uint8* resultPtr, uint32* resultLengthPtr )`	
Service ID[hex]:	0x72	
Sync/Async:	Sync or Async, dependend on the job configuration	
Reentrancy:	Reentrant	
Parameters (in):	jobId	Holds the identifier of the job using the CSM service.
Parameters (inout):	resultLengthPtr	Holds a pointer to the memory location in which the result length in bytes is stored. On calling this function, this parameter shall contain the number of random bytes, which shall be stored to the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	resultPtr	Holds a pointer to the memory location which will hold the result of the random number generation.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: request failed, service is still busy CRYPTO_E_QUEUE_FULL: request failed, the queue is full CRYPTO_E_ENTROPY_EXHAUSTION: request failed, entropy of random number generator is exhausted.
Description:	Starts the random number generation service of the CSM module. If the service state is not "idle", the function shall return with "CRYPTO_E_BUSY". Otherwise, this function shall call CryIf_RandomGenerate().	

] (SRS_CryptoStack_00019)

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_RandomGenerate`.

[SWS_Csm_01001] [The `Crypto_JobInfoType` job with the corresponding `jobId` shall be used as parameter in `CryIf_RandomGenerate()` and shall be set in the following way:

```
job->jobPrimitiveInputOutput.outputPtr = resultPtr,
job->jobPrimitiveInputOutput.outputLengthPtr =
resultLengthPtr.
```
] ( )

8.3.9 Key Management Interface

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

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

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

8.3.9.1 Key Setting Interface

8.3.9.1.1 Csm_KeyElementSet

[SWS_Csm_00957] ⌈

Service name:	Csm_KeyElementSet	
Syntax:	`Std_ReturnType Csm_KeyElementSet( uint32 keyId, uint32 keyElementId, const uint8* keyPtr, uint32 keyLength )`	
Service ID[hex]:	0x78	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	keyId	Holds the identifier of the key for which a new material shall be set.
	keyElementId	Holds the identifier of the key element to be written.
	keyPtr	Holds the pointer to the key element bytes to be processed.
	keyLength	Contains the number of key element bytes.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_KEY_WRITE_FAIL:Request failed because write access was denied CRYPTO_E_KEY_NOT_AVAILABLE: Request failed because the key is not available. CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element size does not match size of provided data.
Description:	Sets the given key element bytes to the key identified by keyId.	

⌋ ()

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyElementSet`.

8.3.9.1.2 Csm_KeySetValid

[SWS_Csm_00958] [

Service name:	Csm_KeySetValid	
Syntax:	`Std_ReturnType Csm_KeySetValid( uint32 keyId )`	
Service ID[hex]:	0x67	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	keyId	Holds the identifier of the key for which a new material shall be validated.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: Request Failed, Crypro Driver Object is Busy
Description:	Sets the key state of the key identified by keyId to valid.	

] ()

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeySetValid`.

8.3.9.2 Key Extraction Interface

8.3.9.2.1 Csm_KeyElementGet

[SWS_Csm_00959] [

Service name:	Csm_KeyElementGet	
Syntax:	`Std_ReturnType Csm_KeyElementGet( uint32 keyId, uint32 keyElementId, uint8* keyPtr, uint32* keyLengthPtr )`	
Service ID[hex]:	0x68	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	keyId	Holds the identifier of the key from which a key element shall be extracted.
	keyElementId	Holds the identifier of the key element to be extracted.
Parameters	keyLengthPtr	Holds a pointer to the memory location in which the output buffer

(inout):		length in bytes is stored. On calling this function, this parameter shall contain the buffer length in bytes of the keyPtr. When the request has finished, the actual size of the written input bytes shall be stored.
Parameters (out):	keyPtr	Holds the pointer to the memory location where the key shall be copied to.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CRYPTO_E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_KEY_NOT_AVAILABLE: request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed because read access was denied CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
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.	

] (SRS_CryptoStack_00010, SRS_CryptoStack_00011, SRS_CrytptoStack_00029)

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyElementGet`.

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

8.3.9.2.2 Csm_AsymPublicKeyExtractStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00436] [

Service name:	Csm_AsymPublicKeyExtractStart (obsolete)	
Syntax:	`Std_ReturnType Csm_AsymPublicKeyExtractStart( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x35	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	hold the identifier of the CSM module configuration which has to be used during the key extraction.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated and does nothing. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymPublicKeyExtractStart`.

8.3.9.2.3 Csm_AsymPublicKeyExtractUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00443] [

Service name:	Csm_AsymPublicKeyExtractUpdate (obsolete)	
Syntax:	`Std_ReturnType Csm_AsymPublicKeyExtractUpdate( Csm_ConfigIdType cfgId, const uint8* dataPtr, uint32 dataLength )`	
Service ID[hex]:	0x36	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	dataPtr	holds a pointer to the data which contains the key in a format which cannot be used directly by the CSM. From this data the key will be extracted in a CSM-conforming format
	dataLength	holds the length of the data in bytes
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the asymmetrical public key extraction service with input data. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymPublicKeyExtractUpdate`.

8.3.9.2.4 Csm_AsymPublicKeyExtractFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00450] [

Service name:	Csm_AsymPublicKeyExtractFinish (obsolete)	
Syntax:	`Std_ReturnType Csm_AsymPublicKeyExtractFinish( Csm_ConfigIdType cfgId, Csm_AsymPublicKeyType* keyPtr )`	
Service ID[hex]:	0x37	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	keyPtr	holds a pointer to a structure where the result (i.e. the asymmetrical public key) is stored in
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful

		E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Finishes the asymmetrical public key extraction service. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_AsymPublicKeyExtractFinish`.

8.3.9.2.5 Csm_SymKeyExtractStart

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00418] [

Service name:	Csm_SymKeyExtractStart (obsolete)	

Syntax:	`Std_ReturnType Csm_SymKeyExtractStart(`
	`  Csm_ConfigIdType cfgId`
	`)`	

Service ID[hex]:	0x32	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the key extraction
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated and does nothing. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymKeyExtractStart`.

8.3.9.2.6 Csm_SymKeyExtractUpdate

DEPRECATED: This function will be removed in the next major release!

[SWS_Csm_00425] [

Service name:	Csm_SymKeyExtractUpdate (obsolete)	

Syntax:	`Std_ReturnType Csm_SymKeyExtractUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* dataPtr,`
	`  uint32 dataLength`
	`)`	

Service ID[hex]:	0x33	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to

		be used during the operation.
	dataPtr	holds a pointer to the data which contains the key in a format which cannot be used directly by the CSM. From this data the key will be extracted in a CSM-conforming format
	dataLength	holds the length of the data in bytes
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds the symmetrical key extraction service with input data. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymKeyExtractUpdate`.

8.3.9.2.7 Csm_SymKeyExtractFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00432] [

Service name:	Csm_SymKeyExtractFinish (obsolete)	
Syntax:	`Std_ReturnType Csm_SymKeyExtractFinish( Csm_ConfigIdType cfgId, Csm_SymKeyType* keyPtr )`	
Service ID[hex]:	0x34	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	keyPtr	holds a pointer to a structure where the result (i.e. the symmetrical key) is stored in.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Finishes the symmetrical key extraction service. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_SymKeyExtractFinish`.

8.3.9.3 Key Copying Interface

8.3.9.3.1 Csm_KeyElementCopy

[SWS_Csm_00969] [

Service name:	Csm_KeyElementCopy

Syntax:	`Std_ReturnType Csm_KeyElementCopy( const uint32 keyId, const uint32 keyElementId, const uint32 targetKeyId, const uint32 targetKeyElementId )`	
Service ID[hex]:	0x71	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for the same keyId	
Parameters (in):	keyId	Holds the identifier of the key whose key element shall be the source element.
	keyElementId	Holds the identifier of the key element which shall be the source for the copy operation.
	targetKeyId	Holds the identifier of the key whose key element shall be the destination element.
	targetKeyElementId	Holds the identifier of the key element which shall be the destination for the copy operation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request FailedCRYPTO_E_BUSY: Request Failed, Crypto Driver Object is Busy 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_WRITE_FAIL: Request failed, not allowed to write key element. CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible.
Description:	This function shall copy a key elements from one key to a target key.	

] ()

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyElementCopy`.

8.3.9.3.2 Csm_KeyCopy

[SWS_Csm_01034] [

Service name:	Csm_KeyCopy
Syntax:	`Std_ReturnType Csm_KeyCopy( const uint32 keyId, const uint32 targetKeyId )`

Service ID[hex]:	0x73	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key whose key element shall be the source element.
	targetKeyId	Holds the identifier of the key whose key element shall be the destination element.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed 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.
Description:	This function shall copy all key elements from the source key to a target key.	

] ()

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyCopy`.

8.3.9.4 Key Generation interface

8.3.9.4.1 Csm_RandomSeed

[SWS_Csm_01051] ⌈

Service name:	Csm_RandomSeed	

Syntax:	`Std_ReturnType Csm_RandomSeed(`
	`  uint32 keyId,`
	`  const uint8* seedPtr,`
	`  uint32 seedLength`
	`)`	

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

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	This function shall dispatch the random seed function to the configured crypto driver object.	

] ()

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_RandomSeed`.

8.3.9.4.2 Csm_RandomSeedStart

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00149] ⌈

Service name:	Csm_RandomSeedStart (obsolete)	

Syntax:	`Std_ReturnType Csm_RandomSeedStart(`
	`  Csm_ConfigIdType cfgId`
	`)`	

Service ID[hex]:	0x0c	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the seeding of the random number generator.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated and does nothing. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_RandomSeedStart`.

8.3.9.4.3 Csm_RandomSeedUpdate

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00156] ⌈

Service name:	Csm_RandomSeedUpdate (obsolete)

Syntax:	`Std_ReturnType Csm_RandomSeedUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* seedPtr,`
	`  uint32 seedLength`
	`)`

Service ID[hex]:	0x0d

Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
	seedPtr	holds a pointer to the seed for the random number generator.
	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 CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated. Feeds a seed to the random seed service. **Tags:** atp.Status=obsolete	

] ()

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_RandomSeedUpdate`.

8.3.9.4.4 Csm_RandomSeedFinish

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00163] [

Service name:	Csm_RandomSeedFinish (obsolete)	

Syntax:	`Std_ReturnType Csm_RandomSeedFinish(`
	`  Csm_ConfigIdType cfgId`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy
Description:	This function is deprecated and does nothing. **Tags:** atp.Status=obsolete	

] ()
Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_RandomSeedFinish`.

8.3.9.4.5 Csm_KeyGenerate

[SWS_Csm_00955] [

Service name:	Csm_KeyGenerate

Syntax:	`Std_ReturnType Csm_KeyGenerate(`
	`  uint32 keyId`
	`)`

Service ID[hex]:	0x6a
Sync/Async:	Synchronous

Reentrancy:	Reentrant but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key for which a new material shall be generated.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	Generates new key material and store it in the key identified by keyId.	

] (SRS_CryptoStack_00026, SRS_CryptoStack_00027)

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyGenerate`.

8.3.9.5 Key Derivation Interface

In cryptography, a key derivation function (or KDF) is a function, which derives one or more secret keys from a secret value and/or other known information such as a passphrase or cryptographic key.
Specification of input keys that are protected by hardware means can be achieved by using the Csm_KeyDeriveKey interface.

8.3.9.5.1 Csm_KeyDerive

[SWS_Csm_00956] [

Service name:	Csm_KeyDerive	

Syntax:	`Std_ReturnType Csm_KeyDerive(`
	`  uint32 keyId,`
	`  uint32 targetKeyId`
	`)`	

Service ID[hex]:	0x6b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key which is used for key derivation.
	targetKeyId	Holds the identifier of the key which is used to store the derived key.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy
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.	

] (SRS_CryptoStack_00103)

[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`.
] ( )

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyDerive`.

8.3.9.6 Key Exchange Interface

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

8.3.9.6.1 Csm_KeyExchangeCalcPubVal

[SWS_Csm_00966] [

Service name:	Csm_KeyExchangeCalcPubVal	
Syntax:	`Std_ReturnType Csm_KeyExchangeCalcPubVal( uint32 keyId, uint8* publicValuePtr, uint32* publicValueLengthPtr )`	
Service ID[hex]:	0x6c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
Parameters (inout):	publicValueLengthPtr	Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	publicValuePtr	Contains the pointer to the data where the public value shall be stored.
Return value:	Std_ReturnType	Wrong return values - here are the correct ones: E_OK: request successfulE_NOT_OK: request failed CRYPTO_E_KEY_NOT_VALID: request failed, the key's state is "invalid" CRYPTO_E_SMALL_BUFFER: the provided buffer is too small to store the result.
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.	

] (SRS_CrytptoStack_00028)

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyExchangeCalcPubVal`.

8.3.9.6.2 Csm_KeyExchangeCalcSecret

[SWS_Csm_00967] [

Service name:	Csm_KeyExchangeCalcSecret	
Syntax:	`Std_ReturnType Csm_KeyExchangeCalcSecret( uint32 keyId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength )`	
Service ID[hex]:	0x6d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key which shall be used for the key exchange protocol.
	partnerPublicValuePtr	Holds the pointer to the memory location which contains the partner's public value.
	partnerPublicValueLength	Contains the length of the partner's public value in bytes.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result
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.	

] (SRS_CrytptoStack_00028)

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyExchangeCalcSecret`.

8.3.9.7 Certificate Interface

8.3.9.7.1 Csm_CertificateParse

[SWS_Csm_01036] [

Service name:	Csm_CertificateParse
Syntax:	`Std_ReturnType Csm_CertificateParse( const uint32 keyId`

	)	
Service ID[hex]:	0x6e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for same keyId	
Parameters (in):	keyId	Holds the identifier of the key to be used for the certificate parsing.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	This function shall dispatch the certificate parse function to the CRYIF.	

] (SRS_CrytptoStack_00031)

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_CertificateParse`.

8.3.9.7.2 Csm_CertificateVerify

[SWS_Csm_01038] [

Service name:	Csm_CertificateVerify	
Syntax:	`Std_ReturnType Csm_CertificateVerify( const uint32 keyId, const uint32 verifyCryIfKeyId, Crypto_VerifyResultType* verifyPtr )`	
Service ID[hex]:	0x74	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant but not for the same cryptoKeyId	
Parameters (in):	keyId	Holds the identifier of the key which shall be used to validate the certificate.
	verifyCryIfKeyId	Holds the identifier of the key containing the certificate to be verified.
Parameters (inout):	None	
Parameters (out):	verifyPtr	Holds a pointer to the memory location which will contain the result of the certificate verification.
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId. Note: Only certificates stored in the same Crypto Driver can be verified against each other. If the key element CRYPTO_KE_CERTIFICATE_CURRENT_TIME is used for the verification of the validity period of the certificate indentified by verifyKeyId, it shall have the same format as the timestamp in the certificate.	

] ()

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

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_CertificateVerify`.

8.3.10 Job Cancellation Interface

8.3.10.1 Csm_CancelJob

[SWS_Csm_00968] ⌈

Service name:	Csm_CancelJob	
Syntax:	`Std_ReturnType Csm_CancelJob( uint32 job, Crypto_OperationModeType mode )`	
Service ID[hex]:	0x6f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	job	Holds the identifier of the job to be canceled
	mode	Not used, just for interface compatibility provided.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed
Description:	Removes the job in the Csm Queue and calls the job's callback with the result CRYPTO_E_JOB_CANCELED. It also passes the cancellation command to the CryIf to try to cancel the job in the Crypto Driver.	

⌋ ()

[SWS_Csm_01021] ⌈ If no errors are detected by Csm, the service `Csm_CancelJob()` shall remove the job from the job queue and shall call `CryIf_CancelJob()`.
⌋ ( )

[SWS_Csm_01030] ⌈ If a job is removed from the job queue, the service `Csm_CancelJob()` shall call the callback of the job with the return value `CRYPTO_E_JOB_CANCELED`.
⌋ ( )

Regarding error detection, the requirement **SWS_Csm_00489** is applicable to the function `Csm_KeyExchangeCalcPubVal`.

8.3.11 Callback Notifications

8.3.11.1 Csm_CallbackNotification

[SWS_Csm_00970] ⌈

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

⌋ (SRS_BSW_00359, SRS_BSW_00360)

[SWS_Csm_01053]⌈If the `CRYPTO_OPERATIONMODE_UPDATE` bit is set in `job->jobPrimitiveInputOutput.mode` and the corresponding `CsmJobPrimitiveCallbackUpdateNotification` (**ECUC_CSM_00064**) is true, the `Csm_CallbackFunction` shall call the configured callback function.

⌋()

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

⌋()

8.3.11.2 Csm_<Service>CallbackNotification

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00455] ⌈

Service name:	Csm_<Service>CallbackNotification	
Syntax:	`void Csm_<Service>CallbackNotification( Std_ReturnType Result )`	
Service ID[hex]:	0x79	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Result	Contains the result of the cryptographic operation
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	This function is deprecated. This function shall call the callback function as given in the configuration of the service <Service> with the argument given by "Result".

] (SRS_BSW_00359, SRS_BSW_00360)

8.3.11.3 Csm_<Service>ServiceFinishNotification

DEPRECATED: This function will be removed in the next major release!
[SWS_Csm_00457] [

Service name:	Csm_<Service>ServiceFinishNotification (obsolete)

Syntax:	`void Csm_<Service>ServiceFinishNotification(`
	`  void`
	`)`

Service ID[hex]:	0x75
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This function is deprecated and does nothing. **Tags:** atp.Status=obsolete

] (SRS_BSW_00359, SRS_BSW_00360)

8.3.12 Scheduled functions

8.3.12.1 Csm_MainFunction

[SWS_Csm_00479] [

Service name:	Csm_MainFunction

Syntax:	`void Csm_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x01
Description:	API to be called cyclically to process the requested jobs. The Csm_MainFunction shall check the queues for jobs to pass to the underlying CRYIF.

] (SRS_BSW_00373, SRS_BSW_00432)

8.4 Expected Interfaces

8.4.1 Interfaces to Standard Software Modules

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

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

[SWS_Csm_00486] ⌈The CSM module shall use an AUTOSAR Dem module to report errors to the DEM.
⌋()

8.5 Mandatory Interfaces

API function	*Description*

8.6 Optional Interfaces

API function	*Description*

8.7 Service Interface

This chapter is an addition to the specification of the Csm module. Whereas the other parts of the specification define the behavior and the C-interfaces of the corresponding basic software module, this chapter formally specifies the corresponding AUTOSAR service in terms of the SWC template. The interfaces described here will be visible on the VFB and are used to generate the RTE between application software and the Csm module.

8.7.1 Client-Server-Interfaces

8.7.1.1 CsmKeyManagement_{Config}

[SWS_Csm_01905] ⌈

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

	6	CSM_E_KEY_READ_FAIL
	7	CSM_E_KEY_WRITE_FAIL
	8	CSM_E_KEY_NOT_AVAILABLE
	10	CSM_E_KEY_SIZE_MISMATCH

Operations

CertificateParse		
Comments	This function shall dispatch the certificate parse function to the CRYIF.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

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

KeyCopy			
Comments	This function shall copy all key elements from the source key to a target key.		
Variation	--		
Parameters	targetKeyId	Comment	Holds the identifier of the key whose key element shall be the destination element.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_SIZE_MISMATCH	The service 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.	

KeyDerive			
Comments	Derives a new key by using the key elements in the given key. The given key contains the key elements for the password and salt. The derived key is stored in the key element with the id 1 of the key identified by targetCryptoKeyId.		
Variation	--		
Parameters	targetKeyId	Comment	Holds the identifier of the key which is used to store the derived key.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

KeyElementCopy			
Comments	This function shall copy a key elements from one key to a target key		
Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element which shall be the source for the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyId	Comment	Holds the identifier of the key whose key element shall be the destination element.
		Type	uint32
		Variation	--
		Direction	IN
	targetKeyElementId	Comment	Holds the identifier of the key element which shall be the destination for the copy operation.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_SIZE_MISMATCH	The service 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.	

KeyElementGet

Comments	Retrieves the key element bytes from a specific key element of the key and stores the key element in the provided buffer.		
Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element to be read.
		Type	uint32
		Variation	--
		Direction	IN
	keyPtr	Comment	Holds the data to the key element bytes to be written.
		Type	Csm_KeyDataType_{Crypto}
		Variation	tbd
		Direction	OUT
	keyLength	Comment	Contains the number of key element bytes.
		Type	uint32
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	
	CSM_E_KEY_READ_FAIL	The service request failed because read access was denied.	
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.	

KeyElementSet			
Comments	Sets the given key element bytes to the key.		
Variation	--		
Parameters	keyElementId	Comment	Holds the identifier of the key element to be written.
		Type	uint32
		Variation	--

		Direction	IN
	keyPtr	Comment	Holds the data to the key element bytes to be processed.
		Type	Csm_KeyDataType_{Crypto}
		Variation	tbd
		Direction	IN
	keyLength	Comment	Contains the number of key element bytes.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	
	CSM_E_KEY_WRITE_FAIL	The service request failed because write access was denied.	
	CSM_E_KEY_NOT_AVAILABLE	The service request failed because the key is not available.	
	CSM_E_KEY_SIZE_MISMATCH	The service request failed because the key element is not partially accessible and the provided key element length is too short or too long for that key element.	

KeyExchangeCalcPubVal			
Comments	Calculates the public value of the current user for the key exchange and stores the public key in the provided buffer		
Variation	--		
Parameters	publicValuePtr	Comment	Holds a pointer to the memory location in which the public value length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
		Type	Csm_PublicValueDataType_{Crypto}
		Variation	--

		Direction	OUT
	publicValueLengthPtr	Comment	Contains the pointer to the data where the public value shall be stored.
		Type	uint32
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

KeyExchangeCalcSecret			
Comments	Calculates the shared secret key for the key exchange with the key material of the key identified by the keyId and the partner public key. The shared secret key is stored as a key element in the same key.		
Variation	--		
Parameters	partnerPublicValuePtr	Comment	Holds the pointer to the memory location containing the partner's public value
		Type	Csm_PublicValueDataType_{Crypto}
		Variation	--
		Direction	IN
	partnerPublicValueLength	Comment	Contains the number of bytes of the partner pulic value
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

KeyGenerate		
Comments	Generates new key material and store it in the key identified by keyId.	
Variation	--	
Possible	E_OK	Operation successful

Errors	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

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

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

] (SRS_Csm_00066)

8.7.1.2 **CsmHash_{Config}**

[SWS_Csm_00946] [

Name	CsmHash_{Primitive}
Comment	Interface to execute the hash calculation.

IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

Hash			
Comments	Streaming approach of the hash calculation.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be hashed.
		Type	Csm_HashDataType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmHash/CsmHashConfig.SHORT-NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be hashed.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the hash.
		Type	Csm_HashResultType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmHash/CsmHashConfig.SHORT-NAME)}
		Direction	INOUT

	resultLength	Comment	Contains the length in bytes of the hash.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_CryptoStack_00090)

8.7.1.3 **CsmHash**

DEPRECATED: This Interface will be removed in the next major release!
[SWS_Csm_00775] [

Name	CsmHash (obsolete)	
Comment	Interface to execute the hash calculation. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

HashFinish (obsolete)			
Comments	This Operation is deprecated. Finishes the hash service and stores the hash. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	resultBuffer	Comment	Contains the data of the hash.
		Type	HashResultBuffer
		Variation	--

		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the hash.
		Type	uint32
		Variation	--
		Direction	INOUT
	TruncationIsAllowed	Comment	Indicates if the truncation of the hash is allowed or not.
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

HashStart (obsolete)			
Comments	-- **Tags:** atp.Status=obsolete		
Variation	--		
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

HashUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the hash service with the input data. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be hashed.
		Type	HashDataBuffer
		Variation	--
		Direction	IN

	dataLength	Comment	Contains the length in bytes of the data to be hashed.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

] (SRS_Csm_00066)

8.7.1.4 **CsmMacGenerate_{Config}**

[SWS_Csm_009000] [

Name	CsmMacGenerate_{Primitive}	
Comment	Interface to execute the MAC generation.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmMacGenerate/CsmMacGenerateConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
MacGenerate		
Comments	Uses the given data to perform a MAC generation and stores the MAC in the memory	

	location pointed to by the MAC pointer.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
		Type	Csm_MacGenerateDataType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmMacGenerate/CsmMacGenerateConfig. SHORT-NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the MAC.
		Type	Csm_MacGenerateResultType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmMacGenerate/CsmMacGenerateConfig. SHORT-NAME)}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the MAC.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_CryptoStack_00090)

8.7.1.5 **CsmMacGenerate**

DEPRECATED: This Interface will be removed in the next major release!
[SWS_Csm_00776] [

Name	CsmMacGenerate (obsolete)

Comment	Interface to execute the MAC generation. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

MacGenerateFinish (obsolete)			
Comments	This operation is deprecated. Finishes the MAC generation service and stores the MAC. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	resultBuffer	Comment	Contains the data of the MAC.
		Type	MacGenerateResultBuffer
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the MAC.
		Type	uint32
		Variation	--
		Direction	INOUT
	TruncationIsAllowed	Comment	Indicates if the truncation of the mac shall be truncated or not.
		Type	boolean
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

MacGenerateStart (obsolete)			
Comments	This operationis deprecated. Sets the key for MAC generation. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_SymKeyType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

MacGenerateUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the MAC generate service with the input data. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data from which a MAC shall be generated of.
		Type	MacGenerateDataBuffer
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

] (SRS_Csm_00066)

8.7.1.6 CsmMacVerify_{Config}

[SWS_Csm_00936] ⌈

Name	CsmMacVerify_{Primitive}	
Comment	Interface to execute the MAC verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

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

MacVerify			
Comments	Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the length in bytes of the data from which a MAC shall be generated of.
		Type	Csm_MacVerifyDataType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig.SHORT-NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for whichs MAC shall be verified.
		Type	uint32

		Variation	--
		Direction	IN
	compareBuffer	Comment	Contains the MAC to be verified.
		Type	Csm_MacVerifyCompareType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig.SHORT-NAME)}
		Direction	IN
	compareLength	Comment	Contains the length in BITS of the MAC to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the MAC.
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_CryptoStack_00090)

8.7.1.7 **CsmMacVerify**

DEPRECATED: This Interface will be removed in the next major release!
[SWS_Csm_00777] [

Name	CsmMacVerify (obsolete)	
Comment	Interface to execute the MAC verification. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY

Operations

MacVerifyFinish (obsolete)			
Comments	This function is deprecated. Finishes the MAC verification and stores the verification result. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	MacBuffer	Comment	Contains the MAC to be verified.
		Type	MacVerifyCompareBuffer
		Variation	--
		Direction	IN
	MacLength	Comment	Contains the length in BITS of the MAC to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the result of the MAC verification.
		Type	Csm_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

MacVerifyStart (obsolete)			
Comments	This operation is deprecated. Sets the key for MAC verification. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_SymKeyType
		Variation	--

		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

MacVerifyUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the MAC verification service with the input data. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data for whichs MAC shall be verified.
		Type	MacVerifyDataBuffer
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for whichs MAC shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

] (SRS_Csm_00066)

8.7.1.8 **CsmEncrypt_{Config}**

[SWS_Csm_00947] [

Name	CsmEncrypt_{Primitive}	
Comment	Interface to execute the encryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmEncrypt/CsmEncryptConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

CancelJob			
Comments	Cancels the job.		
Variation	--		
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
Encrypt			
Comments	Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be encrypted.
		Type	Csm_EncryptDataType_{Crypto}
		Variation	Crypto ={ecuc(Csm/CsmPrimitives/ CsmEncrypt/CsmEncryptConfig.SHORT-NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	result	Comment	Contains the data of the cipher.
		Type	Csm_EncryptResultType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmEncrypt/CsmEncryptConfig.SHORT-NAME)}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the cipher.
		Type	uint32

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

] (SRS_CryptoStack_00906)

8.7.1.9 **CsmDecrypt_{Config}**

[SWS_Csm_01906] [

Name	CsmDecrypt_{Primitive}	
Comment	Interface to execute the decryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

Decrypt			
Comments	Streaming approach of the decryption.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data to be decrypted.
		Type	Csm_DecryptDataType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/

			CsmDecrypt/CsmDecryptConfig.SHORT-NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data to be decrypted.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the decrypted plaintext.
		Type	Csm_DecryptResultType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmDecrypt/CsmDecryptConfig.SHORT-NAME)}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the decrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_CryptoStack_00090)

8.7.1.10 CsmSymBlockEncrypt

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00780] [

Name	CsmSymBlockEncrypt (obsolete)	
Comment	Interface to execute the symmetric block encryption. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymBlockEncryptFinish (obsolete)			
Comments	This operation is deprecated. Finish the symmetrical block encrypt service. **Tags:** atp.Status=obsolete		
Variation	--		
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SymBlockEncryptStart (obsolete)			
Comments	This operation is deprecated. Sets the key for symmetrical block encryption. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_SymKeyType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SymBlockEncryptUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the symmetrical block encrypt service with the input data and stores the ciphertext. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	plainTextBuffer	Comment	This operation is deprecated. Feeds the symmetrical block encrypt service with the

			input data and stores the ciphertext.
		Type	SymBlockEncryptDataBuffer
		Variation	--
		Direction	IN
	plainTextLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	cipherTextBuffer	Comment	Contains the data of the cipher.
		Type	SymBlockEncryptResultBuffer
		Variation	--
		Direction	OUT
	cipherTextLength	Comment	Contains the data of the cipher.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_Csm_00066)

8.7.1.11 CsmSymBlockDecrypt

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00781] [

Name	CsmSymBlockDecrypt (obsolete)	
Comment	Interface to execute the symmetric block decryption. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymBlockDecryptFinish (obsolete)			
Comments	This operation is deprecated. Finishes the symmetrical block decrypt service. **Tags:** atp.Status=obsolete		
Variation	--		
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SymBlockDecryptStart (obsolete)			
Comments	This operation is deprecated. Sets the key for symmetrical block decryption. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_SymKeyType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SymBlockDecryptUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the symmetrical block decrypt service with the input data and store the decrypted plaintext. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	cipherTextBuffer	Comment	Contains the data to be decrypted.

		Type	SymBlockDecryptDataBuffer
		Variation	--
		Direction	IN
	cipherTextLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	plainTextBuffer	Comment	Contains the data of the encrypted plaintext.
		Type	SymBlockDecryptResultBuffer
		Variation	--
		Direction	OUT
	plainTextLength	Comment	Contains the length in bytes of the data of the encrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_Csm_00066)

8.7.1.12 CsmSymEncrypt

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00782] [

Name	CsmSymEncrypt (obsolete)	
Comment	Interface to execute the symmetric encryption. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymEncryptFinish (obsolete)			
Comments	This operation is deprecated. Finish the symmetrical encrypt service. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	cipherTextBuffer	Comment	Contains the data of the cipher.
		Type	SymEncryptResultBuffer
		Variation	--
		Direction	OUT
	cipherTextLength	Comment	Contains the length in bytes of the data of the cipher.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

SymEncryptStart (obsolete)			
Comments	This operation is deprecated. Sets the key for symmetrical encryption. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_SymKeyType
		Variation	--
		Direction	IN

	InitVectorBuffer	Comment	Contains the data of the initiation vector.
		Type	SymEncryptInitVectorBuffer
		Variation	--
		Direction	IN
	InitVectorLength	Comment	Contains the length in bytes of the data of the initiation vector.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SymEncryptUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the symmetrical encrypt service with the input data and stores the ciphertext. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	plainTextBuffer	Comment	Contains the data to be encrypted.
		Type	SymEncryptDataBuffer
		Variation	--
		Direction	IN
	plainTextLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	cipherTextBuffer	Comment	Contains the data of the cipher.
		Type	SymEncryptResultBuffer
		Variation	--
		Direction	OUT
	cipherTextLength	Comment	Contains the length in bytes of the cipher.

		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_Csm_00066)

8.7.1.13 CsmSymDecrypt

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00783] [

Name	CsmSymDecrypt (obsolete)	
Comment	Interface to execute the symmetric decryption. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymDecryptFinish			
Comments	This operation is deprecated. Finishes the symmetrical decrypt service.		
Variation	--		
Parameters	plainTextBuffer	Comment	Contains the data of the encrypted plaintext.
		Type	SymDecryptResultBuffer
		Variation	--
		Direction	OUT
	plainTextLength	Comment	Contains the length in bytes of the data of

			the encrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

SymDecryptStart (obsolete)			
Comments	This operation is deprecated. Sets the key for symmetrical decryption. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_SymKeyType
		Variation	--
		Direction	IN
	InitVectorBuffer	Comment	Contains the data of the initiation vector.
		Type	SymDecryptInitVectorBuffer
		Variation	--
		Direction	IN
	InitVectorLength	Comment	Contains the length in bytes of the data of the initiation vector.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SymDecryptUpdate (obsolete)

Comments	This operation is deprecated. Feeds the symmetrical decrypt service with the input data and store the decrypted plaintext. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	cipherTextBuffer	Comment	Contains the data to be decrypted
		Type	SymDecryptDataBuffer
		Variation	--
		Direction	IN
	cipherTextLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	plainTextBuffer	Comment	Contains the data of the encrypted plaintext.
		Type	SymDecryptResultBuffer
		Variation	--
		Direction	OUT
	plainTextLength	Comment	Contains the length in bytes of the data of the encrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_Csm_00066)

8.7.1.14 CsmAsymEncrypt

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00784] [

Name	CsmAsymEncrypt (obsolete)

Comment	Interface to execute the asymmetric encryption. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

AsymEncryptFinish (obsolete)			
Comments	This operation is deprecated. Finish the asymmetrical encrypt service. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	cipherTextBuffer	Comment	Contains the data of the cipher.
		Type	AsymEncryptResultBuffer
		Variation	--
		Direction	OUT
	cipherTextLength	Comment	Contains the length in bytes of the data of the cipher.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

AsymEncryptStart (obsolete)	
Comments	This operation is deprecated. Sets the key for asymmetrical encryption. **Tags:** atp.Status=obsolete

Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_AsymPublicKeyType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

AsymEncryptUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the asymmetrical encrypt service with the input data and stores the ciphertext. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	plainTextBuffer	Comment	Contains the data to be encrypted.
		Type	AsymEncryptDataBuffer
		Variation	--
		Direction	IN
	plainTextLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--
		Direction	IN
	cipherTextBuffer	Comment	Contains the data of the cipher.
		Type	AsymEncryptResultBuffer
		Variation	--
		Direction	OUT
	cipherTextLength	Comment	Contains the length in bytes of the data of the cipher.
		Type	uint32
		Variation	--
		Direction	INOUT

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result

] (SRS_Csm_00066)

8.7.1.15 CsmAsymDecrypt

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00785] [

Name	CsmAsymDecrypt (obsolete)	
Comment	Interface to execute the asymmetric decryption. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

AsymDecryptFinish (obsolete)			
Comments	This operation is deprecated. Finishes the symmetrical decrypt service. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	plainTextBuffer	Comment	Contains the data of the encrypted plaintext.
		Type	AsymDecryptResultBuffer
		Variation	--
		Direction	OUT
	plainTextLength	Comment	Contains the length in bytes of the data of the encrypted plaintext.
		Type	uint32

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

AsymDecryptStart (obsolete)			
Comments	This operation is deprecated. Sets the key for asymmetrical decryption. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	--
		Type	Csm_AsymPrivateKeyType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

AsymDecryptUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the symmetrical decrypt service with the input data and store the decrypted plaintext. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	cipherTextBuffer	Comment	Contains the data to be decrypted.
		Type	AsymDecryptDataBuffer
		Variation	--
		Direction	IN
	cipherTextLength	Comment	Contains the length in bytes of the data to be encrypted.
		Type	uint32
		Variation	--

		Direction	IN
	plainTextBuffer	Comment	Contains the data of the encrypted plaintext.
		Type	AsymDecryptResultBuffer
		Variation	--
		Direction	OUT
	plainTextLength	Comment	Contains the length in bytes of the data of the encrypted plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_Csm_00066)

8.7.1.16 CsmAEADEncrypt_{Config}

[SWS_Csm_01910] [

Name	CsmAEADEncrypt_{Primitive}	
Comment	Interface to execute the AEAD encryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

AEADEncrypt	
Comments	Streaming approach of the AEAD encryption.
Variation	--

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

	macBuffer	Comment	Contains the data of the MAC.
		Type	Csm_AEADEncryptMacType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig.SHORT-NAME)}
		Direction	OUT
	macLength	Comment	Contains the length in bytes of the data of the MAC.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

CancelJob		
Comments	Cancels the job.	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

] (SRS_CryptoStack_00090)

8.7.1.17 CsmAEADDecrypt_{Config}

[SWS_Csm_01915] ⌈

Name	CsmAEADDecrypt_{Primitive}	
Comment	Interface to execute the AEAD decryption.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig.SHORT-NAME)}	
Possible	0	E_OK

Errors	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

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

		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig.SHORT-NAME)}
		Direction	IN
	macLength	Comment	Contains the length in BITS of the data of the MAC.
		Type	uint32
		Variation	--
		Direction	IN
	plaintextBuffer	Comment	Contains the data of the decrypted AEAD plaintext.
		Type	Csm_AEADDecryptPlaintextType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig.SHORT-NAME)}
		Direction	OUT
	plaintextLength	Comment	Contains the length in bytes of the data of the decrypted AEAD plaintext.
		Type	uint32
		Variation	--
		Direction	INOUT
	verifyPtr	Comment	--
		Type	Crypto_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

CancelJob	
Comments	Cancels the job.
Variation	--

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

] (SRS_CryptoStack_00090)

8.7.1.18 CsmSignatureGenerate_{Config}

[SWS_Csm_00903] [

Name	CsmSignatureGenerate_{Primitive}	
Comment	--	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/ CsmSignatureGenerateConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

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

SignatureGenerate			
Comments	Streaming approach of the signature generation.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the length in bytes of the data from which the signature shall be generated.
		Type	Csm_SignatureGenerateDataType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmSignatureGenerate/ CsmSignatureGenerateConfig.SHORT-

			NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which the signature shall be generated.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the signature.
		Type	Csm_SignatureGenerateResultType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/ CsmSignatureGenerate/ CsmSignatureGenerateConfig.SHORT-NAME)}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the signature.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_CryptoStack_00090)

8.7.1.19 CsmSignatureGenerate

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00786] [

Name	CsmSignatureGenerate (obsolete)	
Comment	Interface to execute the signature generation. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SignatureGenerateFinish (obsolete)			
Comments	This operation is deprecated. Finishes the signature generation service and stores the signature. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	resultBuffer	Comment	Contains the signature.
		Type	SignatureGenerateResultBuffer
		Variation	--
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the signature.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

SignatureGenerateStart (obsolete)			
Comments	This operationis deprecated. Sets the key for signature generation. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	Identifier of the key.
		Type	Csm_AsymPrivateKeyType
		Variation	--
		Direction	IN

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

SignatureGenerateUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the signature generate service with the input data. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data from which a signature shall be generated of.
		Type	SignatureGenerateDataBuffer
		Variation	--
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data from which a signaure shall be generated of.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

] (SRS_Csm_00066)

8.7.1.20 CsmSignatureVerify_{Config}

[SWS_Csm_00943] [

Name	CsmSignatureVerify_{Primitive}	
Comment	Interface to execute the signature verification.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmSignatureVerify/ CsmSignatureVerifyConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

CancelJob			
Comments	Cancels the job.		
Variation	--		
SignatureVerify			
Comments	Streaming approach of the signature verification.		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data for whichs signature shall be verified.
		Type	Csm_SignatureVerifyDataType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmSignatureVerify/CsmSignatureVerifyConfig.SHORT-NAME)}
		Direction	IN
	dataLength	Comment	Contains the length in bytes of the data for whichs signature shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
	compareBuffer	Comment	Contains the signature to be verified.
		Type	Csm_SignatureVerifyCompareType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmSignatureVerify/CsmSignatureVerifyConfig.SHORT-NAME)}
		Direction	IN
	compareLength	Comment	Contains the length in bytes of the signature to be verified.
		Type	uint32
		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the data of the random number.
		Type	Crypto_VerifyResultType

		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

] (SRS_CryptoStack_00090)

8.7.1.21 CsmSignatureVerify

DEPRECATED: This interface will be removed in the next major release!
[SWS_Csm_00787] [

Name	CsmSignatureVerify (obsolete)	
Comment	Interface to execute the signature verification. **Tags:** atp.Status=obsolete	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

SignatureVerifyFinish (obsolete)			
Comments	This function is deprecated. Finishes the signature verification and stores the verification result. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	signatureBuffer	Comment	Contains the signature to be verified.
		Type	SignatureVerifyCompareSignatureBuffer
		Variation	--
		Direction	IN
	signatureLength	Comment	Contains the length in bytes of the signature to be verified.
		Type	uint32

		Variation	--
		Direction	IN
	resultBuffer	Comment	Contains the result of the signature verification.
		Type	Csm_VerifyResultType
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SignatureVerifyStart (obsolete)			
Comments	This operation is deprecated. Sets the key for signature verification. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	key	Comment	This operation is deprecated. Sets the key for signature verification.
		Type	Csm_AsymPublicKeyType
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

SignatureVerifyUpdate (obsolete)			
Comments	This operation is deprecated. Feeds the signature verification service with the input data. **Tags:** atp.Status=obsolete		
Variation	--		
Parameters	dataBuffer	Comment	Contains the data for whichs signature shall be verified.
		Type	SignatureVerifyDataBuffer
		Variation	--
		Direction	IN

	dataLength	Comment	Contains the length in bytes of the data for whichs signature shall be verified.
		Type	uint32
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	

] (SRS_Csm_00066)

8.7.1.22 CsmSecureCounter

[SWS_Csm_09260] [

Name	CsmSecureCounter_{Primitive}	
Comment	Interface to return the secure counter.	
IsService	true	
Variation	Primitive = {ecuc(Csm/CsmPrimitives/CsmSecureCounter/ CsmSecureCounterConfig.SHORT-NAME)}	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SecureCounterIncrement			
Comments	Increments the secure counter.		
Variation	--		
Parameters	stepSize	Comment	Contains the value by which the counter will be incremented
		Type	uint64
		Variation	--
		Direction	IN
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	

	CSM_E_BUSY	failed, service is still busy
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result

SecureCounterRead			
Comments	Returns the secure counter.		
Variation	--		
Parameters	counterValuePtr	Comment	Contains the value of the secure counter
		Type	uint64
		Variation	--
		Direction	OUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_SMALL_BUFFER	the provided buffer is too small to store the result	

] (SRS_CryptoStack_00090)

8.7.1.23 CsmRandomGenerate_{Config}

[SWS_Csm_00902] [

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

Operations

RandomGenerate	
Comments	Contains the length in bytes of the data of random number.

Variation	--		
Parameters	resultBuffer	Comment	Contains the random number
		Type	Csm_RandomGenerateResultType_{Crypto}
		Variation	Crypto = {ecuc(Csm/CsmPrimitives/CsmRandomGenerate/CsmRandomGenerateConfig.SHORT-NAME)}
		Direction	OUT
	resultLength	Comment	Contains the length in bytes of the data of random number.
		Type	uint32
		Variation	--
		Direction	INOUT
Possible Errors	E_OK	Operation successful	
	E_NOT_OK	--	
	CSM_E_BUSY	failed, service is still busy	
	CSM_E_ENTROPY_EXHAUSTION	request failed, entropy of random number generator is exhausted.	

] (SRS_CryptoStack_00090)

8.7.1.24 CallbackNotification

[SWS_Csm_00928] [

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

Operations

CallbackNotification			
Comments	Notifies the application with a return value that the job has finished.		
Variation	--		
Parameters	job	Comment	--

		Type	Crypto_JobType
		Variation	--
		Direction	IN
	result	Comment	Return value that shall be returned to the application
		Type	Csm_ResultType
		Variation	--
		Direction	IN

] (SRS_CryptoStack_00090)

8.7.2 Implementation Data Types

8.7.2.1 Crypto_JobStateType

[SWS_Csm_01028] [

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

] ()

8.7.2.2 Crypto_ServiceInfoType

[SWS_Csm_01031] [

Name	Crypto_ServiceInfoType		
Kind	Enumeration		
Range	CRYPTO_HASH	0x00	Hash Service
	CRYPTO_MACGENERATE	0x01	MacGenerate Service
	CRYPTO_MACVERIFY	0x02	MacVerify Service

	CRYPTO_ENCRYPT	0x03	Encrypt Service
	CRYPTO_DECRYPT	0x04	Decrypt Service
	CRYTPO_AEADENCRYPT	0x05	AEADEncrypt Service
	CRYPTO_AEADDECRYPT	0x06	AEADDecrypt Service
	CRYPTO_SIGNATUREGENERATE	0x07	SignatureGenerate Service
	CRYPTO_SIGNATUREVERIFY	0x08	SignatureVerify Service
	CRYPTO_SECCOUNTERINCREMENT	0x09	SecureCounterIncrement Service
	CRYPTO_SECCOUNTERREAD	0x0A	SecureCounterDecrement Service
	CRYPTO_RANDOMGENERATE	0x0B	RandomGenerate Service
Description	Enumeration of the kind of the service.		
Variation	--		

] ()

8.7.2.3 **Crypto_AlgorithmModeType**

[SWS_Csm_01048] [

Name	Crypto_AlgorithmModeType		
Kind	Enumeration		
Range	CRYPTO_ALGOMODE_NOT_SET	0x00	Algorithm key is not set
	CRYPTO_ALGOMODE_ECB	0x01	Blockmode: Electronic Code Book
	CRYPTO_ALGOMODE_CBC	0x02	Blockmode: Cipher Block Chaining
	CRYPTO_ALGOMODE_CFB	0x03	Blockmode: Cipher Feedback Mode
	CRYPTO_ALGOMODE_OFB	0x04	Blockmode: Output Feedback Mode
	CRYPTO_ALGOMODE_CTR	0x05	Blockmode: Counter Modex
	CRYPTO_ALGOMODE_GCM	0x06	Blockmode: Galois/Counter Mode
	CRYPTO_ALGOMODE_XTS	0x07	XOR-encryption-based tweaked-codebook mode with ciphertext stealing
	CRYPTO_ALGOMODE_RSAES_OAEP	0x08	RSA Optimal Asymmetric Encryption Padding
	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5	0x09	RSA encryption/decryption

			with PKCS#1 v1.5 padding
	CRYPTO_ALGOMODE_RSASSA_PSS	0x0a	RSA Probabilistic Signature Scheme
	CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5	0x0b	RSA signature with PKCS#1 v1.5
	CRYPTO_ALGOMODE_8ROUNDS	0x0c	8 rounds (e.g. ChaCha8)
	CRYPTO_ALGOMODE_12ROUNDS	0x0d	12 rounds (e.g. ChaCha12)
	CRYPTO_ALGOMODE_20ROUNDS	0x0e	20 rounds (e.g. ChaCha20)
	CRYPTO_ALGOMODE_HMAC	0x0f	Hashed-based MAC
	CRYPTO_ALGOMODE_CMAC	0x10	Cipher-based MAC
	CRYPTO_ALGOMODE_GMAC	0x11	Galois MAC
	CRYPTO_ALGOMODE_CTRDRBG	0x12	Counter-based Deterministic Random Bit Generator
	CRYPTO_ALGOMODE_SIPHASH_2_4	0x13	Siphash-2-4
	CRYPTO_ALGOMODE_SIPHASH_4_8	0x14	Siphash-4-8
	CRYPTO_ALGOMODE_CUSTOM	0xff	Custom algorithm mode
Description	Enumeration of the algorithm mode		
Variation	--		

] ()

8.7.2.4 **Crypto_AlgorithmInfoType**

[SWS_Csm_01008] [

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

] ()

8.7.2.5 Crypto_ProcessingType

[SWS_Csm_01049] [

Name	Crypto_ProcessingType		
Kind	Enumeration		
Range	CRYPTO_PROCESSING_ASYNC	0x00	Asynchronous job processing
	CRYPTO_PROCESSING_SYNC	0x01	Synchronous job processing
Description	Enumeration of the processing type.		
Variation	--		

] (SRS_CryptoStack_00100, SRS_CryptoStack_00101)

8.7.2.6 Crypto_VerifyResultType

[SWS_Csm_01024] [

Name	Crypto_VerifyResultType		
Kind	Enumeration		
Range	CRYPTO_E_VER_OK	0x00	The result of the verification is "true", i.e. the two compared elements are identical. This return code shall be given as value "0"
	CRYPTO_E_VER_NOT_OK	0x01	The result of the verification is "false", i.e. the two compared elements are not identical. This return code shall be given as value "1".
Description	Enumeration of the result type of verification operations.		
Variation	--		

] ()

8.7.2.7 Crypto_JobPrimitiveInputOutputType

[SWS_Csm_01009] [

Name	Crypto_JobPrimitiveInputOutputType		
Kind	Structure		
Elements	inputPtr	uint8*	Pointer to the input data.
	inputLength	Crypto_AlgorithmModeType	Contains the input length in bytes.
	secondaryInputPtr	const uint8*	Pointer to the secondary

			input data (for MacVerify, SignatureVerify).
	secondaryInputLength	uint32	Contains the secondary input length in bytes.
	tertiaryInputPtr	const uint8*	Pointer to the tertiary input data (for MacVerify, SignatureVerify).
	tertiaryInputLength	uint32	Contains the tertiary input length in bytes.
	outputPtr	uint8*	Pointer to the output data.
	outputLengthPtr	uint32*	Holds a pointer to a memory location containing the output length in bytes.
	secondaryOutputPtr	uint8*	Pointer to the secondary output data.
	secondaryOutputLengthPtr	uint32*	Holds a pointer to a memory location containing the secondary output length in bytes.
	verifyPtr	Crypto_VerifyResultType*	Output pointer to a memory location holding a Crypto_VerifyResultType
	output64Ptr	uint64*	Output pointer to a memory location holding an uint64.
	mode	Crypto_OperationModeType	Indicator of the mode(s)/operation(s) to be performed
Description	Structure which contains input and output information depending on the job and the crypto primitive.		
Variation	--		

] ()

8.7.2.8 **Crypto_JobInfoType**

[SWS_Csm_01010] [

Name	Crypto_JobInfoType		
Kind	Structure		
Elements	jobId	const uint32	The family of the algorithm
	jobPriority	const uint32	Specifies the importance of the job (the higher, the more important).

Description	Structure which contains job information (job ID and job priority).
Variation	--

] (SRS_CryptoStack_00102)

8.7.2.9 Crypto_PrimitiveInfoType

[SWS_Csm_01011] [

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

] ()

8.7.2.10 Crypto_JobPrimitiveInfoType

[SWS_Csm_01012] [

Name	Crypto_JobPrimitiveInfoType		
Kind	Structure		
Elements	callbackId	const uint32	Identifier of the callback function, to be called, if the configured service finished.
	primitiveInfo	const Crypto_PrimitiveInfoType*	Pointer to a structure containing further configuration of the crypto primitives
	secureCounterId	const uint32	Identifier of a secure counter.
	cryIfKeyId	const uint32	Identifier of the CryIf key.
	processingType	const boolean	Determines the synchronous or asynchronous behavior.
	callbackUpdateNotification	const Crypto_ProcessingType	Indicates, whether the callback function shall be called, if the UPDATE operation has finished.

Description	Structure which contains further information, which depends on the job and the crypto primitive.
Variation	--

] (SRS_CryptoStack_00008)

8.7.2.11 Crypto_JobType

[SWS_Csm_01013] [

Name	Crypto_JobType		
Kind	Structure		
Elements	jobId	const uint32	Identifier for the job structure.
	state	Crypto_JobStateType	Determines the current job state.
	PrimitiveInputOutput	Crypto_JobPrimitiveInputOutputType	Structure containing input and output information depending on the job and the crypto primitive.
	jobPrimitiveInfo	const Crypto_JobPrimitiveInfoType*	Pointer to a structure containing further information, which depends on the job and the crypto primitive
	jobInfo	const Crypto_JobInfoType*	Pointer to a structure containing further information, which depends on the job and the crypto primitive
	cryptoKeyId	uint32	Identifier of the Crypto Driver key. The identifier shall be written by the Crypto Interface
Description	Structure which contains further information, which depends on the job and the crypto primitive.		
Variation	--		

] ()

8.7.2.12 Csm_KeyDataType_{Crypto}

[SWS_Csm_00828] [

Name	Csm_KeyDataType_{Crypto}		
Kind	Array	Element type	uint8

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

] ()

8.7.2.13 Csm_SeedDataType_{Crypto}

[SWS_Csm_00829] [

Name	Csm_SeedDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	sum({ecuc(Csm/CsmKeys/CsmKey/CsmKeyRef->CryIfKey/CryIfKeyRef->CryptoKey/CryptoKeyTypeRef->CryptoKeyType/CryptoKeyElementRef->CryptoKeyElement/CryptoKeyElementSize) Elements		
Description	Array long enough to store the entropy. Array size is depending on the underlying RNG		
Variation	Crypto = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}		

] (SRS_CryptoStack_00090)

8.7.2.14 Csm_PublicValueDataType_{Crypto}

[SWS_Csm_00827] [

Name	Csm_PublicValueDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	sum({ecuc(Csm/CsmKeys/CsmKey/CsmKeyRef->CryIfKey/CryIfKeyRef->CryptoKey/CryptoKeyTypeRef->CryptoKeyType/CryptoKeyElementRef->CryptoKeyElement/CryptoKeyElementSize) Elements		
Description	Array long enough to store the public value. Array size is depending on the underlying algorithm		
Variation	Crypto = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}		

] (SRS_CryptoStack_00090)

8.7.2.15 Csm_ResultType

[SWS_Csm_91001] [

Name	Csm_ResultType
Kind	Type
Derived	Std_ReturnType

from			
Description	Csm module specific return values for use in Std_ReturnType that could occur on async.		
Range	E_SMALL_BUFFER	0x02	The service request failed because the provided buffer is too small to store the result.
	E_ENTROPY_EXHAUSTION	0x03	The service request failed because the entropy of random number generator is exhausted.
	E_KEY_READ_FAIL	0x04	The service request failed because read access was denied.
	E_KEY_NOT_AVAILABLE	0x05	The service request failed because the key is not available.
	E_KEY_NOT_VALID	0x06	The service request failed because key was not valid.
	E_JOB_CANCELED	0x07	The service request failed because the job was canceled
Variation	--		

] (SRS_CryptoStack_00095)

8.7.2.16 Csm_ConfigIdType

[SWS_Csm_00691] [

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

] (SRS_Csm_00066)

8.7.2.17 Csm_VerifyResultType

[SWS_Csm_00075] [

Name	Csm_VerifyResultType (obsolete)
Kind	Type

Derived from	uint8		
Description	Enumeration of the result type of verification operations. **Tags:** atp.Status=obsolete		
Range	CSM_E_VER_OK	0	the result of the verification is "true", i.e. the two compared elements are identical. This return code shall be given as value "0"
	CSM_E_VER_NOT_OK	1	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".
Variation	--		

] (SRS_Csm_00066)

8.7.2.18 Csm_AsymPublicKeyType

[SWS_Csm_00076] [

Name	Csm_AsymPublicKeyType		
Kind	Structure		
Elements	length	uint32	This element contains the length in bytes of the key stored in element 'data'
	data	Csm_AsymPublicKeyArrayType	This element contains the key data or a key handle.
Description	Structure for the public asymmetrical key.		
Variation	--		

] (SRS_CryptoStack_00090)

8.7.2.19 Csm_AsymPublicKeyArrayType

[SWS_Csm_00077] [

Name	Csm_AsymPublicKeyArrayType		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmGeneral/CsmAsymPublicKeyMaxLength)} Elements		
Description	Array long enough to store an asymmetric public key.		
Variation	--		

] (SRS_CryptoStack_00090)

8.7.2.20 Csm_AsymPrivateKeyType

[SWS_Csm_01080] [

Name	Csm_AsymPrivateKeyType		
Kind	Structure		
Elements	length	uint32	This element contains the length of the key stored in element 'data'
	data	Array of Csm_AlignType	This element contains the key data or a key handle.
		Size	CSM_ASYM_PRIV_KEY_MAX_SIZE
Description	Structure for the private asymmetrical key. CSM_ASYM_PRIV_KEY_MAX_SIZE shall be chosen such that "CSM_ASYM_PRIV_KEY_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmAsymDecryptMaxKeySize, CsmSignatureGenerateMaxKeySize, CsmAsymPrivateKeyExtractMaxKeySize, CsmAsymPrivateKeyWrapSymMaxPrivKeySize, CsmAsymPrivateKeyWrapAsymMaxPrivKeySize and CsmAsymPrivateKeyUpdateMaxKeySize.		
Variation	--		

] (SRS_CryptoStack_00090)

8.7.2.21 Csm_AsymPrivateKeyArrayType

[SWS_Csm_00081] [

Name	Csm_AsymPrivateKeyArrayType		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmGeneral/CsmAsymPrivateKeyMaxLength)} Elements		
Description	Array long enough to store an asymmetric private key.		
Variation	--		

] (SRS_CryptoStack_00090)

8.7.2.22 Csm_SymKeyType

[SWS_Csm_01082] [

Name	Csm_SymKeyType		
Kind	Structure		
Elements	length	uint32	This element contains the length in bytes of the key stored in element 'data'
	data	Csm_SymKeyArrayType	This element contains the key data or a key handle.

Description	Structure for the symmetrical key.
Variation	--

] (SRS_CryptoStack_00090)

8.7.2.23 Csm_SymKeyArrayType

[SWS_Csm_00083] [

Name	Csm_SymKeyArrayType		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmGeneral/CsmSymKeyMaxLength)} Elements		
Description	Array long enough to store a symmetric key.		
Variation	--		

] (SRS_CryptoStack_00090)

8.7.2.24 Csm_HashDataType_{Crypto}

[SWS_Csm_01920] [

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

] (SRS_CryptoStack_00090)

8.7.2.25 Csm_HashResultType_{Crypto}

[SWS_Csm_00912] [

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

] (SRS_CryptoStack_00090)

8.7.2.26 Csm_HashDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00856] [

Name	HashDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data of the hash calculation. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.27 Csm_HashResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00857] [

Name	HashResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output data of the hash calculation. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.28 Csm_MacGenerateDataType_{Crypto}

[SWS_Csm_00935] [

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

] (SRS_CryptoStack_00090)

8.7.2.29 Csm_MacGenerateResultType_{Crypto}

[SWS_Csm_00927] [

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

] (SRS_CryptoStack_00090)

8.7.2.30 Csm_MacGenerateDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00864] [

Name	MacGenerateDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data of the MAC generation. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.31 Csm_MacGenerateResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00865] [

Name	MacGenerateResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result of the MAC generation. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.32 Csm_MacVerifyDataType_{Crypto}

[SWS_Csm_00802] [

Name	Csm_MacVerifyDataType_{Crypto}		
Kind	Array	Element type	uint8

Size	{ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig/ CsmMacVerifyDataMaxLength} Elements
Description	Array long enough to store the data for whichs MAC shall be verified.
Variation	Crypto= {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig.SHORT-NAME)}

] (SRS_CryptoStack_00090)

8.7.2.33 Csm_MacVerifyCompareType_{Crypto}

[SWS_Csm_00803] [

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

] (SRS_CryptoStack_00090)

8.7.2.34 Csm_MacVerifyDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00867] [

Name	MacVerifyDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for which the verification shall be verified. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.35 Csm_MacVerifyCompareBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00866] [

Name	MacVerifyCompareBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the signature to be verified.		

	Tags: atp.Status=obsolete
Variation	--

] (SRS_Csm_00066)

8.7.2.36 Csm_EncryptDataType_{Crypto}

[SWS_Csm_01921] [

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

] (SRS_CryptoStack_00090)

8.7.2.37 Csm_EncryptResultType_{Crypto}

[SWS_Csm_01922] [

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

] (SRS_CryptoStack_00090)

8.7.2.38 Csm_DecryptDataType_{Crypto}

[SWS_Csm_01923] [

Name	Csm_DecryptDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig/ CsmDecryptDataMaxLength} Elements		
Description	Array long enough to store the data to be decrypted.		
Variation	Crypto= {ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig.SHORT-NAME)}		

] (SRS_CryptoStack_00090)

8.7.2.39 Csm_DecryptResultType_{Crypto}

[SWS_Csm_01924] [

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

] (SRS_CryptoStack_00090)

8.7.2.40 Csm_SymBlockEncryptDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00877] [

Name	SymBlockEncryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for symmetrical block encryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.41 Csm_SymBlockEncryptResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00877] [

Name	SymBlockEncryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for symmetrical block encryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.42 Csm_SymBlockDecryptDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00875] [

Name	SymBlockDecryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for symmetrical block decryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.43 Csm_SymBlockDecryptResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00876] [

Name	SymBlockDecryptResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result for symmetrical block decryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.44 Csm_SymEncryptDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00881] [

Name	SymEncryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for symmetrical encryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.45 Csm_SymEncryptInitVectorBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00882] [

Name	SymEncryptInitVectorBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for symmetrical encryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.46 Csm_SymEncryptResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00883] [

Name	SymEncryptResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result for symmetrical encryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.47 Csm_SymDecryptDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00878] [

Name	SymDecryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for symmetrical decryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.48 Csm_SymDecryptInitVectorBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00879] [

Name	SymDecryptInitVectorBuffer (obsolete)		
Kind	Array	Element type	uint8

Description	Buffer for the input initial vector for symmetrical decryption. **Tags:** atp.Status=obsolete
Variation	--

] (SRS_Csm_00066)

8.7.2.49 Csm_SymDecryptResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00880] [

Name	SymDecryptResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result for symmetrical decryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.50 Csm_AsymEncryptDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00842] [

Name	AsymEncryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for asymmetrical encryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.51 Csm_AsymEncryptResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00843] [

Name	AsymEncryptResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result for asymmetrical encryption. **Tags:** atp.Status=obsolete		

Variation	--

] (SRS_Csm_00066)

8.7.2.52 Csm_AsymDecryptDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00840] [

Name	AsymDecryptDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data for asymmetrical decryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.53 Csm_AsymDecryptResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00841] [

Name	AsymDecryptResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result for asymmetrical decryption. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.54 Csm_AEADEncryptPlaintextType_{Crypto}

[SWS_Csm_01925] [

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

] (SRS_CryptoStack_00090)

8.7.2.55 Csm_AEADEncryptAssociatedDataType_{Crypto}

[SWS_Csm_01928] ⌈

Name	Csm_AEADEncryptAssociatedDataType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig/ CsmAEADEncryptAssociatedDataMaxLength} Elements		
Description	Array long enough to store the data of the header.		
Variation	Crypto= {ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig. SHORT-NAME)}		

⌋ (SRS_CryptoStack_00090)

8.7.2.56 Csm_AEADEncryptCiphertextType_{Crypto}

[SWS_Csm_01927] ⌈

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

⌋ (SRS_CryptoStack_00090)

8.7.2.57 Csm_AEADEncryptTagType_{Crypto}

[SWS_Csm_01926] ⌈

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

⌋ (SRS_CryptoStack_00090)

8.7.2.58 Csm_AEADDecryptCiphertextType_{Crypto}

[SWS_Csm_00922] ⌈

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

] (SRS_CryptoStack_00090)

8.7.2.59 Csm_AEADDecryptAssociatedDataType_{Crypto}

[SWS_Csm_00923] [

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

] (SRS_CryptoStack_00090)

8.7.2.60 Csm_AEADDecryptTagType_{Crypto}

[SWS_Csm_01074] [

Name	Csm_AEADEncryptMacType_{Crypto}		
Kind	Array	Element type	uint8
Size	{ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig/ CsmAEADDecryptMacLength}/8 Elements		
Description	Array long enough to store the data of the Tag.		
Variation	Crypto= {ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig. SHORT-NAME)}		

] (SRS_CryptoStack_00090)

8.7.2.61 Csm_AEADDecryptPlaintextType_{Crypto}

[SWS_Csm_01075] [

Name	Csm_AEADDecryptPlaintextType_{Crypto}

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

] (SRS_CryptoStack_00090)

8.7.2.62 Csm_SignatureGenerateDataType_{Crypto}

[SWS_Csm_01083] [

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

] (SRS_CryptoStack_01076)

8.7.2.63 Csm_SignatureGenerateResultType_{Crypto}

[SWS_Csm_01083] [

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

] (SRS_CryptoStack_01076)

8.7.2.64 Csm_SignatureGenerateDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00871] [

Name	SignatureGenerateDataBuffer (obsolete)		
Kind	Array	Element type	uint8

Description	Buffer for the input data of the signature generation. **Tags:** atp.Status=obsolete
Variation	--

] (SRS_Csm_00066)

8.7.2.65 Csm_SignatureGenerateResultBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00872] [

Name	SignatureGenerateResultBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the output result of the signature generation. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.66 Csm_SignatureVerifyDataType_{Crypto}

[SWS_Csm_01078] [

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

](SRS_CryptoStack_00090)

8.7.2.67 Csm_SignatureVerifyCompareType_{Crypto}

[SWS_Csm_01079] [

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

	SHORT-NAME)}

] (SRS_CryptoStack_00090)

8.7.2.68 Csm_SignatureVerifyDataBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00874] [

Name	SignatureVerifyDataBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input data, for which the signature shall be verificated. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.69 Csm_SignatureVerifyCompareSignatureBuffer

DEPRECATED: This type will be removed in the next major release!
[SWS_Csm_00873] [

Name	SignatureVerifyCompareSignatureBuffer (obsolete)		
Kind	Array	Element type	uint8
Description	Buffer for the input signature to be verified. **Tags:** atp.Status=obsolete		
Variation	--		

] (SRS_Csm_00066)

8.7.2.70 Csm_RandomGenerateResultType_{Crypto}

[SWS_Csm_00930] [

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

] (SRS_CryptoStack_00090)

8.7.3 Ports

8.7.3.1 {Config}_KeyManagement

[SWS_Csm_01042] ⌈

Name	{Key}_KeyManagement		
Kind	ProvidedPort	Interface	CsmKeyManagement_{Key}
Description	Port to execute the key management functions.		
Port Defined Argument Value(s)	Type	uint32	
	Value	{ecuc(Csm/CsmKeys/CsmKey/CsmKeyId)}	
Variation	({ecuc(Csm/CsmKeys/CsmKey.CsmKeyUsePort)} == TRUE) && ({ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == FALSE) Key = {ecuc(Csm/CsmKeys/CsmKey.SHORT-NAME)}		

⌋ (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.2 {Primitive}_Hash

[SWS_Csm_00931] ⌈

Name	{Job}_Hash		
Kind	ProvidedPort	Interface	CsmHash_{Primitive}
Description	Port to execute the hash calculation		
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.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmHash)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig.SHORT-NAME)}		

⌋ (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.3 {Name}_Hash

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00821] ⌈

Name	{Name}_Hash (obsolete)

Kind	ProvidedPort	Interface	CsmHash
Description	This port is deprecated. Used to execute the hash calculation with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmPrimitives/CsmHash/CsmHashConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

8.7.3.4 **{Primitive}_MacGenerate**

[SWS_Csm_00932] [

Name	{Job}_MacGenerate		
Kind	ProvidedPort	Interface	CsmMacGenerate_{Primitive}
Description	Port for a job to generate a MAC		
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.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmMacGenerate)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmMacGenerate/CsmMacGenerateConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmMacGenerate/CsmMacGenerateConfig.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.5 **{Name}_MacGenerate**

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00906] [

Name	{Name}_MacGenerate (obsolete)		
Kind	ProvidedPort	Interface	CsmMacGenerate
Description	This port is deprecated. Used to execute the MAC generation with the		

	deprecated client-server interfaces. **Tags:** atp.Status=obsolete	
Port Defined Argument Value(s)	Type	Csm_ConfigIdType
	Value	{ecuc(Csm/CsmPrimitives/CsmMacGenerate/ CsmMacGenerateConfig)}
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name= {ecuc(Csm/CsmPrimitives/CsmMacGenerate/ CsmMacGenerateConfig.SHORT-NAME)}	

] (SRS_Csm_00066)

8.7.3.6 **{Primitive}_MacVerify**

[SWS_Csm_00934] [

Name	{Job}_MacVerify		
Kind	ProvidedPort	Interface	CsmMacVerify_{Primitive}
Description	Port for a job to verify a MAC		
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.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/ CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmMacVerify)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig. SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.7 **{Name}_MacVerify**

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00907] [

Name	{Name}_MacVerify (obsolete)		
Kind	ProvidedPort	Interface	CsmMacVerify
Description	This port is deprecated. Used to execute the MAC verification with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		

Port Defined Argument Value(s)	Type	Csm_ConfigIdType
	Value	{ecuc(Csm/CsmPrimitives/CsmMacVerify/ CsmMacVerifyConfig)}
Variation	({ecuc(Csm/CsmGeneral.CsmUseDeprecated)}== TRUE) Name = {ecuc(Csm/CsmPrimitives/CsmMacVerify/CsmMacVerifyConfig. SHORT-NAME)}	

] (SRS_Csm_00066)

8.7.3.8 {Primitive}_Encrypt

[SWS_Csm_00933] [

Name	{Job}_Encrypt		
Kind	ProvidedPort	Interface	CsmEncrypt_{Primitive}
Description	Port for a job to execute the encryption.		
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.CsmJobUsePort)} == TRUE) &&({ecuc(Csm/ CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmEncrypt)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmEncrypt/CsmEncryptConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmEncrypt/CsmEncryptConfig.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.9 {Primitive}_Decrypt

[SWS_Csm_00825] [

Name	{Job}_Decrypt		
Kind	ProvidedPort	Interface	CsmDecrypt_{Primitive}
Description	Port for a job to execute the decryption.		
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.CsmJobUsePort)} == TRUE) &&({ecuc(Csm/		

	CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmDecrypt)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmDecrypt/CsmDecryptConfig.SHORT-NAME)}

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.10 {Name}_SymBlockEncrypt

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00914] [

Name	{Name}_SymBlockEncrypt (obsolete)		
Kind	ProvidedPort	Interface	CsmSymBlockEncrypt
Description	This port is deprecated. Used to execute the symmetric block encryption with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmSymBlockEncrypt/CsmSymBlockEncryptConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmPrimitives/CsmSymBlockEncrypt/CsmSymBlockEncryptConfig.SHORT-NAME)})		

] (SRS_Csm_00066)

8.7.3.11 {Name}_SymBlockDecrypt

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00913] [

Name	{Name}_SymBlockDecrypt (obsolete)		
Kind	ProvidedPort	Interface	CsmSymBlockDecrypt
Description	This port is deprecated. Used to execute the symmetric block decryption with the deprecated client-server interfaces **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmSymBlockDecrypt/CsmSymBlockDecryptConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmSymBlockDecrypt/CsmSymBlockDecryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

8.7.3.12 {Name}_SymEncrypt

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00916] ⌈

Name	{Name}_SymEncrypt (obsolete)		
Kind	ProvidedPort	Interface	CsmSymEncrypt
Description	This port is deprecated. Used to execute the symmetric block encryption with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmSymEncrypt/ CsmSymEncryptConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmSymEncrypt/CsmSymEncryptConfig.SHORT-NAME)}		

⌋ (SRS_Csm_00066)

8.7.3.13 {Name}_SymDecrypt

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00915] ⌈

Name	{Name}_SymDecrypt (obsolete)		
Kind	ProvidedPort	Interface	CsmSymDecrypt
Description	This port is deprecated. Used to execute the symmetric decryption with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmSymDecrypt/ CsmSymDecryptConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmSymDecrypt/CsmSymDecryptConfig.SHORT-NAME)}		

⌋ (SRS_Csm_00066)

8.7.3.14 {Name}_AsymEncrypt

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00889] ⌈

Name	{Name}_AsymEncrypt (obsolete)

Kind	ProvidedPort	Interface	CsmAsymEncrypt
Description	This port is deprecated. Used to execute the asymmetric encryption with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmAsymEncrypt/ CsmAsymEncryptConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmAsymEncrypt/CsmAsymEncryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

8.7.3.15 {Name}_AsymDecrypt

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00888] [

Name	{Name}_AsymDecrypt (obsolete)		
Kind	ProvidedPort	Interface	CsmAsymDecrypt
Description	This port is deprecated. Used to execute the asymmetric decryption with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmAsymDecrypt/ CsmAsymDecryptConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmAsymDecrypt/CsmAsymDecryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

8.7.3.16 {Primitive}_AEADEncrypt

[SWS_Csm_00832] [

Name	{Job}_AEADEncrypt		
Kind	ProvidedPort	Interface	CsmAEADEncrypt_{Primitive}
Description	Port for a job to execute the AEAD encryption.		
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.CsmJobUsePort)} == TRUE) &&({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmAEADEncrypt)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmAEADEncrypt/CsmAEADEncryptConfig.SHORT-NAME)}	

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.17 {Primitive}_AEADDecrypt

[SWS_Csm_00833] [

Name	{Job}_AEADDecrypt		
Kind	ProvidedPort	Interface	CsmAEADDecrypt_{Primitive}
Description	Port for a job to execute the AEAD decryption		
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.CsmJobUsePort)} == TRUE) &&({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmAEADDecrypt)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmAEADDecrypt/CsmAEADDecryptConfig.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.18 {Primitive}_SignatureGenerate

[SWS_Csm_00834] [

Name	{Job}_SignatureGenerate		
Kind	ProvidedPort	Interface	CsmSignatureGenerate_{Primitive}
Description	Port for a job to execute the signature generation.		
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.CsmJobUsePort)} == TRUE)&&({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmSignatureGenerate)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/CsmSignatureGenerateConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/CsmSignatureGenerateConfig.SHORT-NAME)}	

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.19 {Name}_SignatureGenerate

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00910] [

Name	{Name}_SignatureGenerate (obsolete)		
Kind	ProvidedPort	Interface	CsmSignatureGenerate
Description	This port is deprecated. Used to execute the signature generation with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/CsmSignatureGenerateConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmPrimitives/CsmSignatureGenerate/CsmSignatureGenerateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

8.7.3.20 {Primitive}_SignatureVerify

[SWS_Csm_00835] [

Name	{Job}_SignatureVerify		
Kind	ProvidedPort	Interface	CsmSignatureVerify_{Primitive}
Description	Port for a job to execute the signature verification.		
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.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmSignatureVerify)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmSignatureVerify/CsmSignatureVerifyConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmSignatureVerify/CsmSignatureVerifyConfig.SHORT-NAME)}

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.21 {Name}_SignatureVerify

DEPRECATED: This port will be removed in the next major release!
[SWS_Csm_00911] [

Name	{Name}_SignatureVerify (obsolete)		
Kind	ProvidedPort	Interface	CsmSignatureVerify
Description	This port is deprecated. Used to execute the signature verification with the deprecated client-server interfaces. **Tags:** atp.Status=obsolete		
Port Defined Argument Value(s)	Type	Csm_ConfigIdType	
	Value	{ecuc(Csm/CsmSignatureVerify/CsmSignatureVerifyConfig)}	
Variation	{ecuc(Csm/CsmGeneral.CsmUseDeprecated)} == TRUE Name = {ecuc(Csm/CsmSignatureVerify/CsmSignatureVerifyConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

8.7.3.22 {Primitive}_SecureCounter

[SWS_Csm_00837] [

Name	{Job}_SecureCounter		
Kind	ProvidedPort	Interface	CsmSecureCounter_{Primitive}
Description	Port to access the secure counter		
Port Defined Argument Value(s)	Type	uint32	
	Value	({ecuc(Csm/CsmJobs/CsmJob.CsmJobId)}	
Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/CsmJobs/CsmJob.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmSecureCounter)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmSecureCounter/CsmSecureCounterConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmSecureCounter/CsmSecureCounterConfig.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.23 {Primitive}_RandomGenerate

[SWS_Csm_00838] [

Name	{Job}_RandomGenerate		
Kind	ProvidedPort	Interface	CsmRandomGenerate_{Primitive}
Description	Port to execute the random numer generation		
Port Defined Argument Value(s)	Type	uint32	
	Value	({ecuc(Csm/CsmJobs/CsmJob.CsmJobId)}	
Variation	({ecuc(Csm/CsmJobs/CsmJob.CsmJobUsePort)} == TRUE) && ({ecuc(Csm/CsmJobs/CsmJob.CsmJobPrimitiveRef -> CsmPrimitives/CsmRandomGenerate)} != NULL) Job = {ecuc(Csm/CsmPrimitives/CsmRandomGenerate/CsmRandomGenerateConfig.SHORT-NAME)} Primitive = {ecuc(Csm/CsmPrimitives/CsmRandomGenerate/CsmRandomGenerateConfig.SHORT-NAME)}		

] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

8.7.3.24 {Primitive}_CallbackNotification

[SWS_Csm_00934] [

Name	{Callback}_CallbackNotification		
Kind	RequiredPort	Interface	CallbackNotification
Description	Port for the callback notification.		
Variation	{ecuc(Csm/CsmCallbacks/CsmCallback/CsmCallbackFunc)} != NULL Callback = {ecuc(Csm/CsmCallbacks/CsmCallback/CsmCallbackFunc.SHORT-NAME)}		

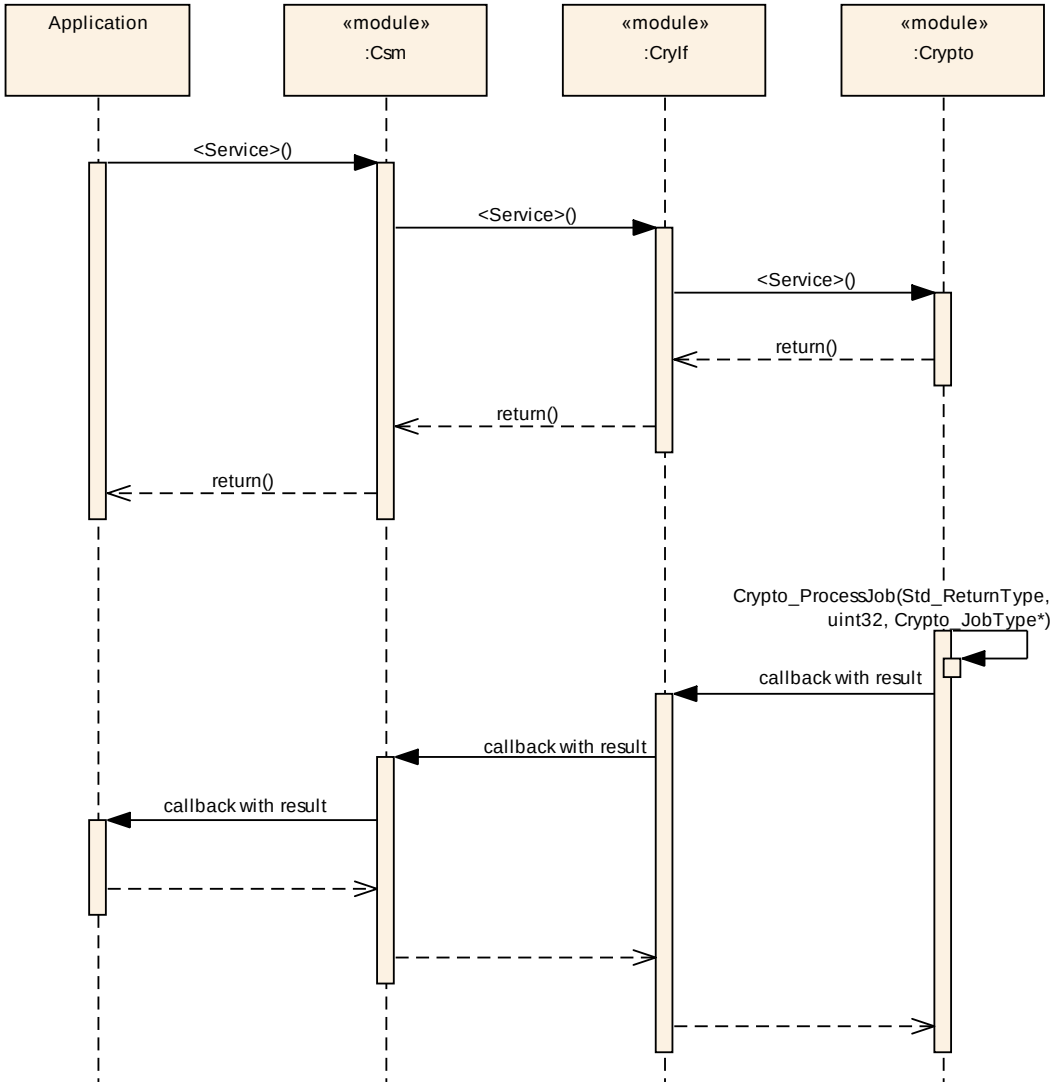
] (SRS_CryptoStack_00090, SRS_CryptoStack_00091)

9 Sequence Diagrams

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

9.1.1 Asynchronous Calls

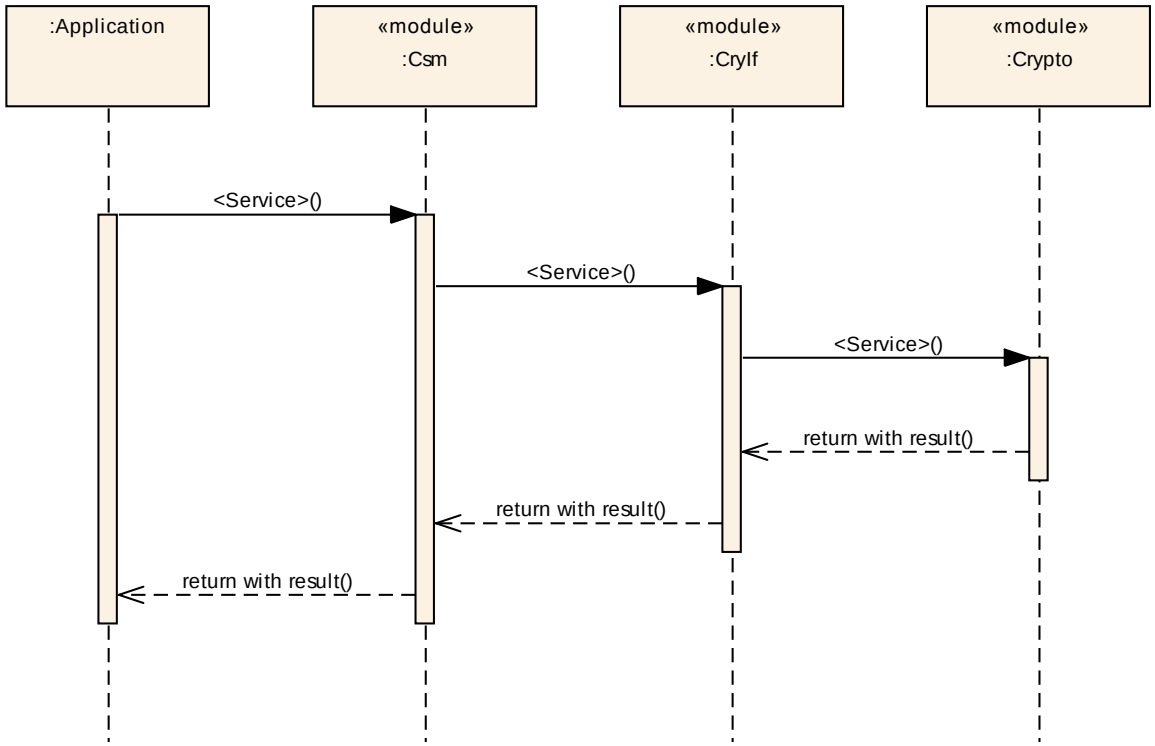
The following diagram (Sequence diagram for asynchronous call) shows a sample sequence of function calls for a request performed asynchronously. The result of the asynchronous function can be accessed after an asynchronous notification (invocation of the configured callback function).



Sequence diagram for asynchronous call with callback

9.1.2 Synchronous Calls

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



Sequence diagram for synchronous call

10 Configuration

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

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

Chapter 10.3 specifies published information of the module CSM.

10.1 How to Read this Chapter

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

10.2 Containers and Configuration Parameters

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

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

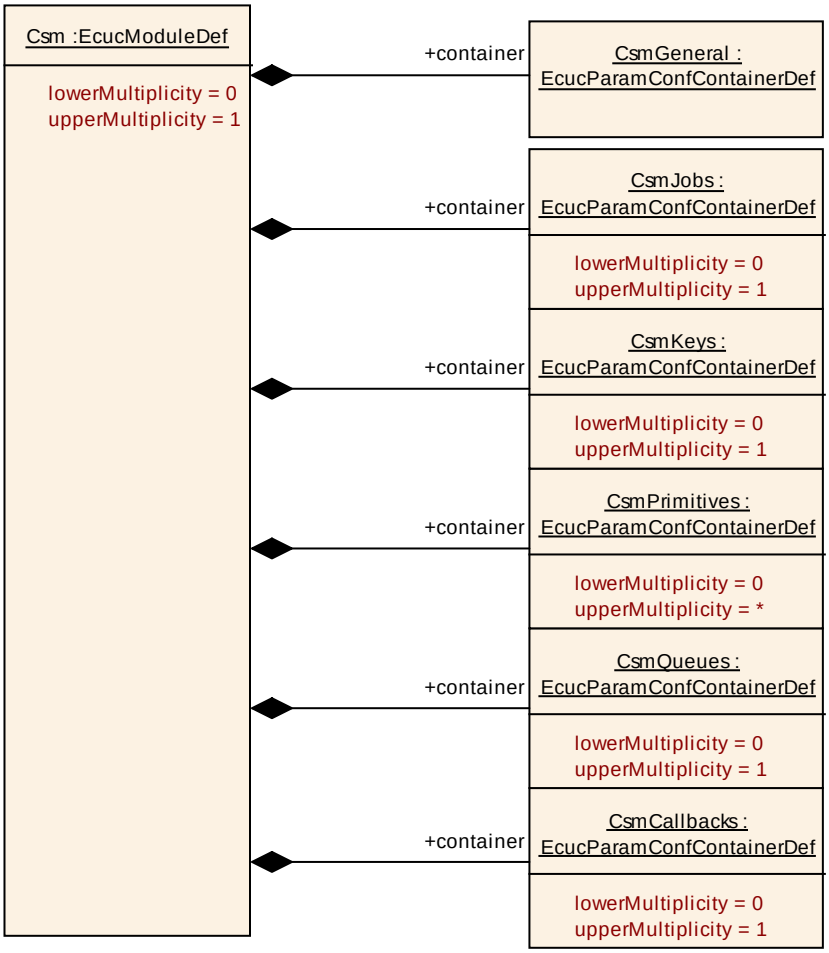


Figure 9-1 Crypto Service Manager Layout

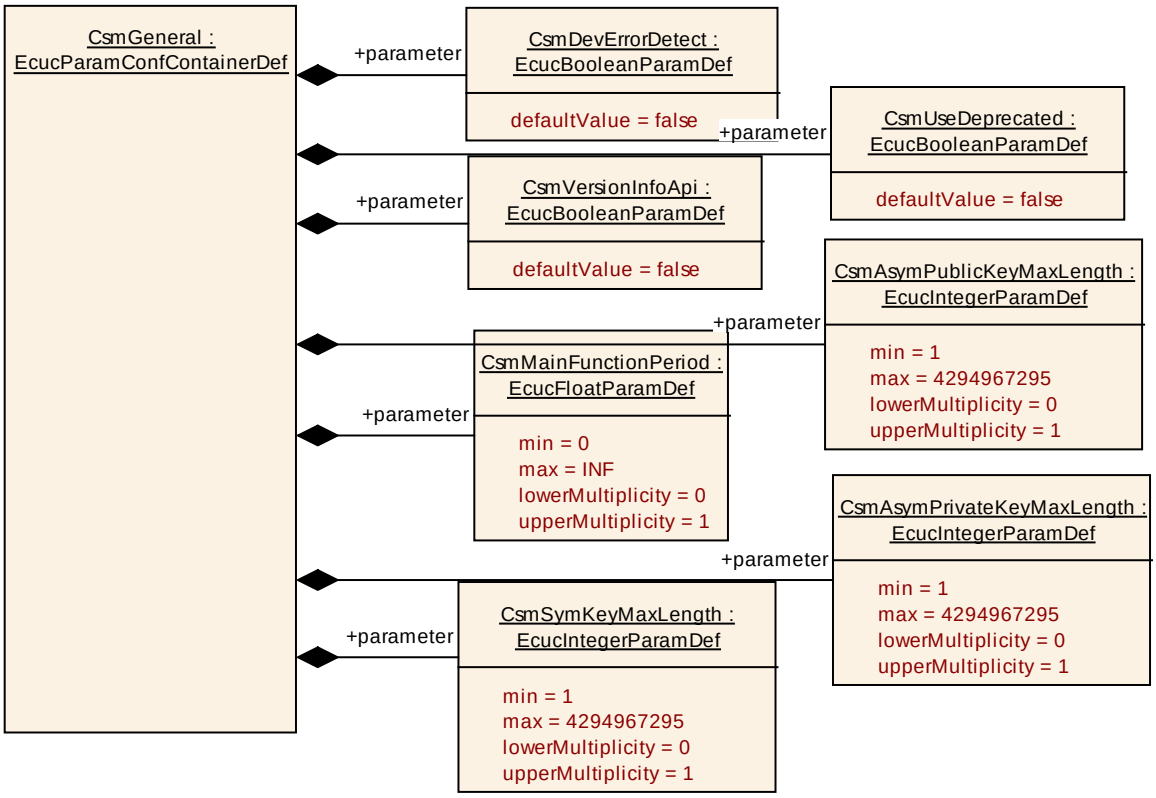


Figure 9-2 Crypto Service Manager General Layout

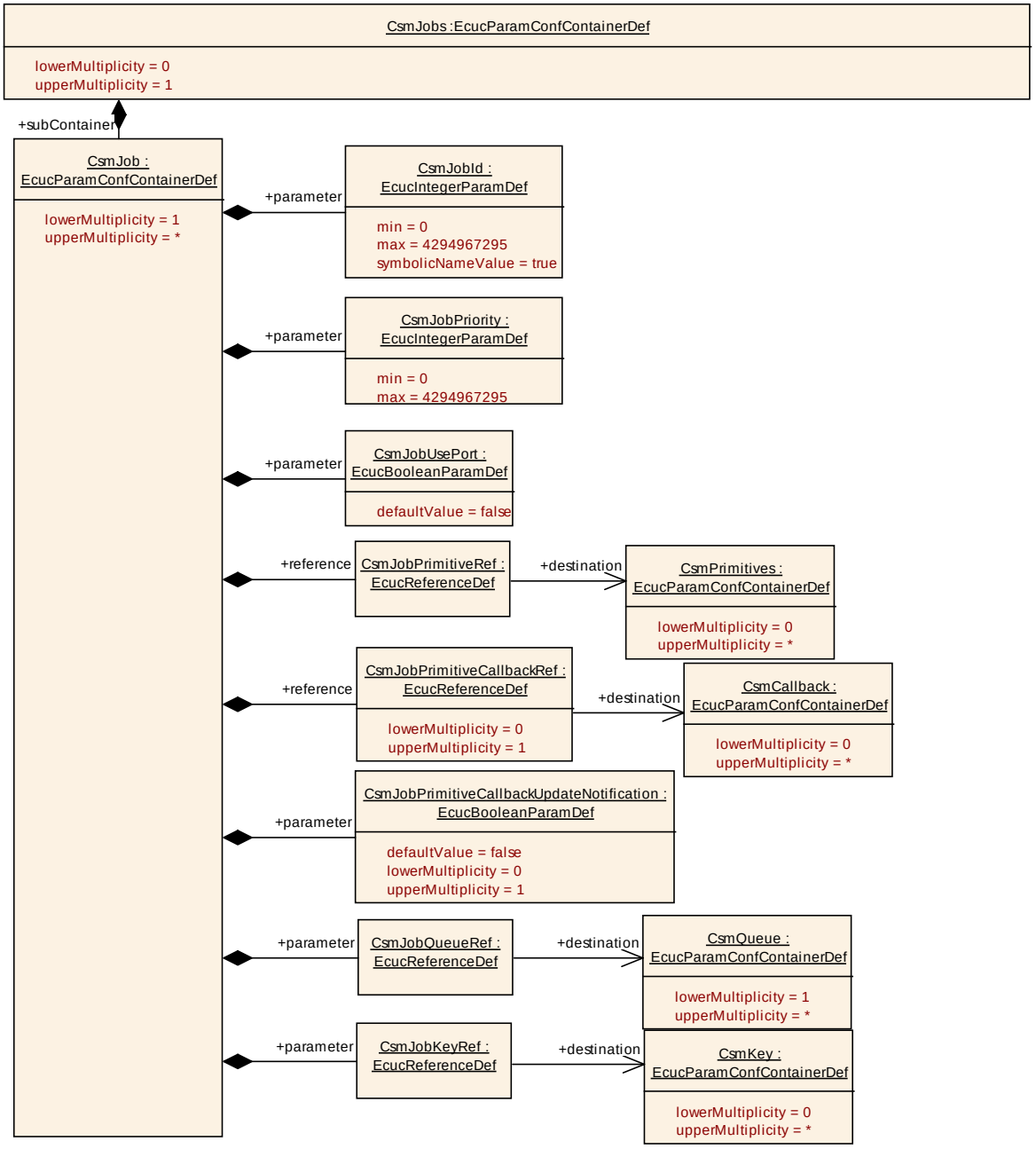


Figure 9-3 Crypto Service Manager Jobs Layout

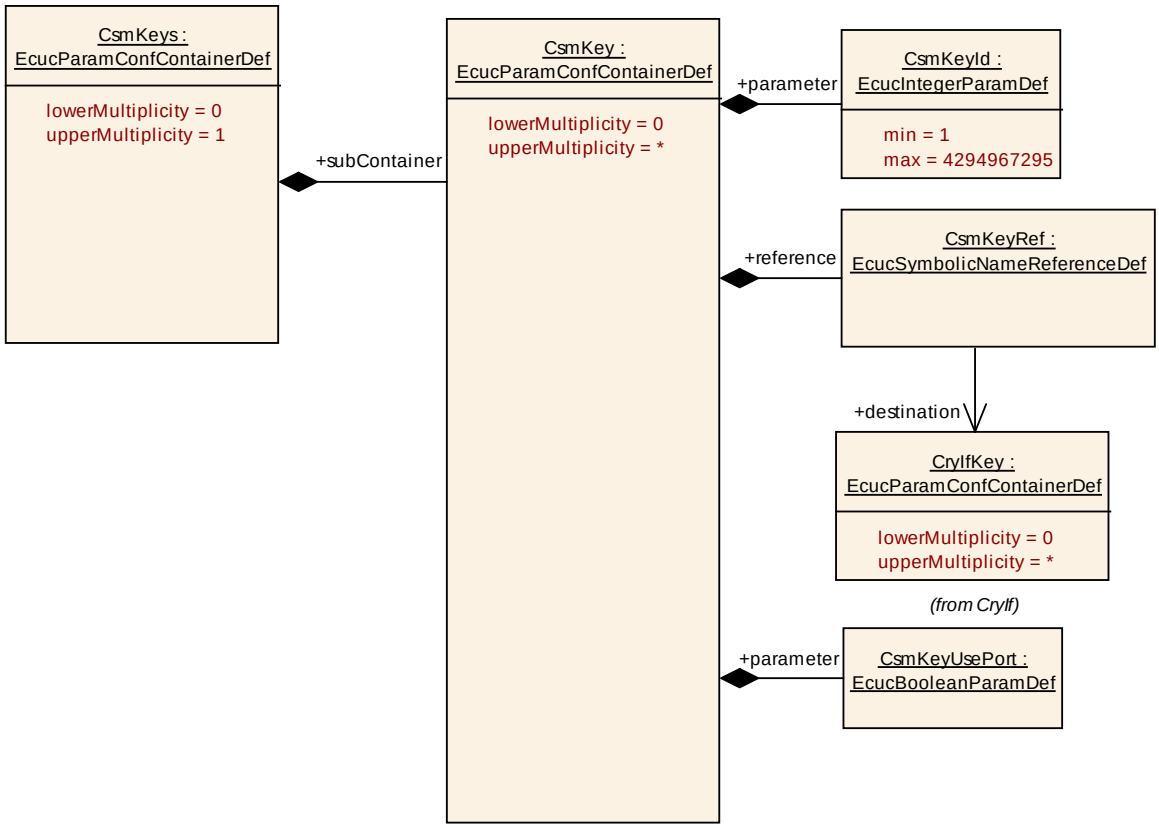


Figure 9-4 Crypto Service Manager Keys Layout

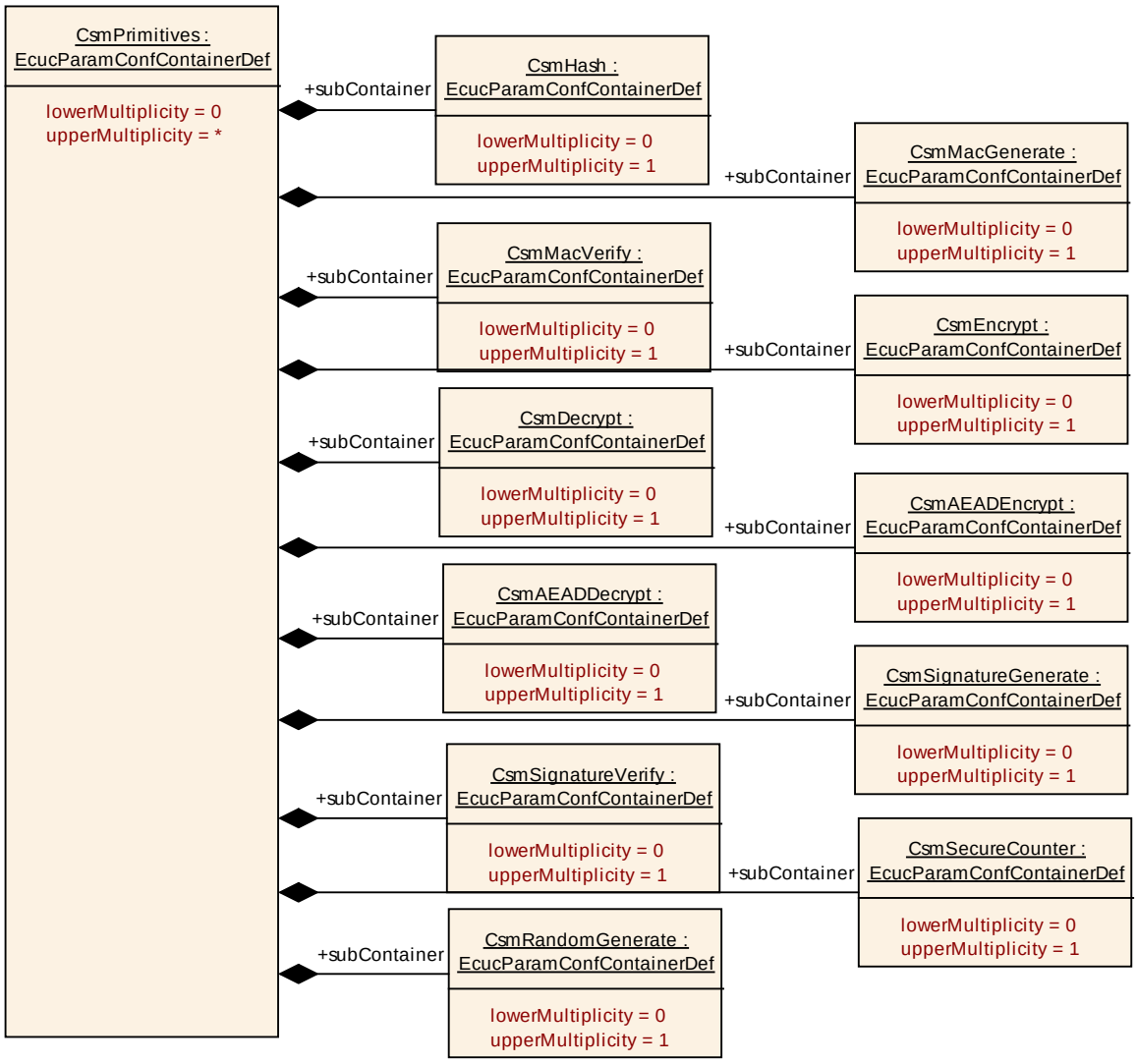


Figure 9-5 Crypto Service Manager Primitives Layout

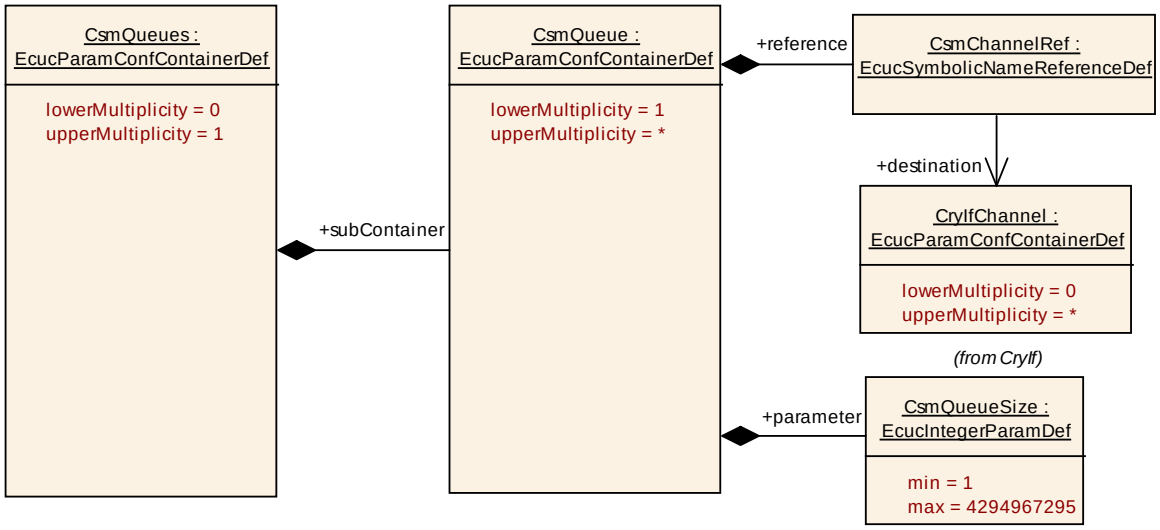


Figure 9-6 Crypto Service Manager Queues Layout

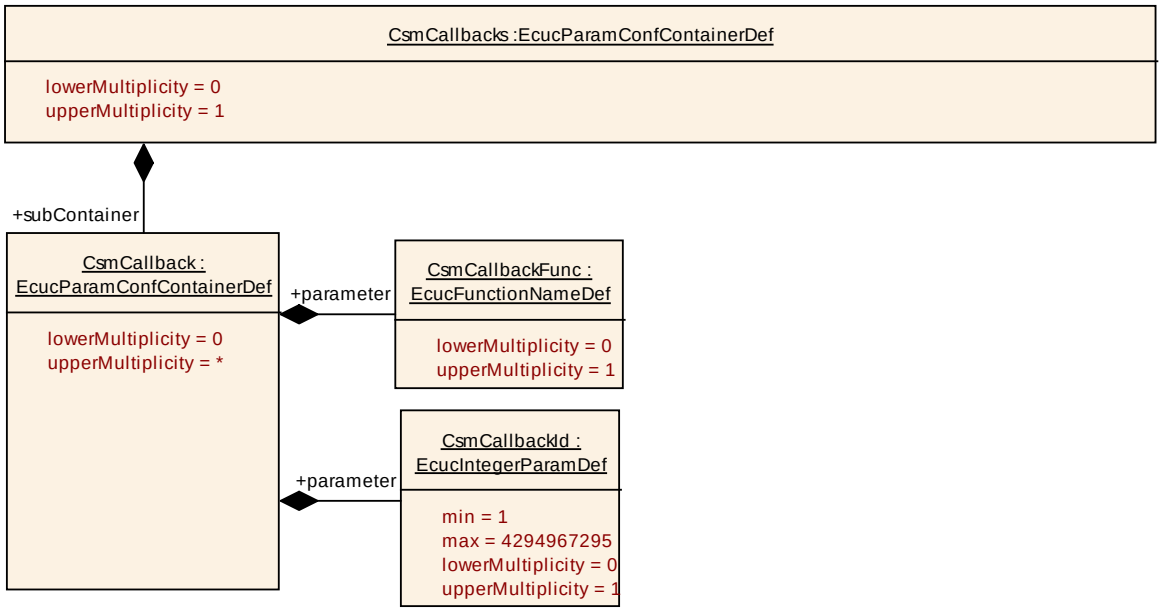


Figure 9-7 Crypto Service Manager Callbacks

10.2.1 Csm

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmCallbacks	0..1	Container for callback function configurations
CsmGeneral	1	Container for common configuration options.
CsmJobs	0..1	Container for configuration of CSM jobs.
CsmKeys	0..1	Container for CSM key configurations.
CsmPrimitives	0..*	Container for configuration of CsmPrimitives
CsmQueues	0..1	Container for CSM queue configurations

10.2.2 CsmGeneral

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

SWS Item	**ECUC_Csm_00115 :**
Name	CsmAsymPrivateKeyMaxLength
Description	Maximum length in bytes of an asymmetric public key for all algorithm

Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

SWS Item	**ECUC_Csm_00001 :**		
Name	CsmDevErrorDetect		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00113 :**		
Name	CsmMainFunctionPeriod		
Description	Specifies the period of main function Csm_MainFunction in seconds.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		

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

SWS Item	**ECUC_Csm_00116 :**		
Name	CsmSymKeyMaxLength		
Description	Maximum length in bytes of a symmetric key for all algorithm		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

SWS Item	**ECUC_Csm_00003 :**		
Name	CsmVersionInfoApi		
Description	Pre-processor switch to enable and disable availability of the API Csm_GetVersionInfo(). True: API Csm_GetVersionInfo() is available. False: API Csm_GetVersionInfo() is not available.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants

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

No Included Containers

10.2.3 **CsmJobs**

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

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

10.2.4 **CsmJob**

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

SWS Item	**ECUC_Csm_00119 :**		
Name	CsmJobId		
Description	Identifier of the CSM job		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00124 :**
Name	CsmJobPrimitiveCallbackUpdateNotification
Description	This parameter indicates, whether the callback function shall be called, if

	the UPDATE operation has been finished.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00120 :**		
Name	CsmJobPriority		
Description	Priority of the job. The higher the value, the higher the job's priority.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00121 :**		
Name	CsmJobUsePort		
Description	Does the job need RTE interfaces? True: the job needs RTE interfaces False: the job needs no RTE interfaces		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

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

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

SWS Item	**ECUC_Csm_00122 :**		
Name	CsmJobPrimitiveRef		
Description	This parameter refers to the used CsmPrimitive. Different jobs may refer to one CsmPrimitive. The referred CsmPrimitive provides detailed information on the actual cryptographic routine.		
Multiplicity	1		
Type	Reference to [CsmPrimitives]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00125 :**		
Name	CsmJobQueueRef		
Description	This parameter refers to the queue. The queue is used if the underlying crypto driver object is busy. The queue refers also to the channel which is used.		
Multiplicity	1		
Type	Reference to [CsmQueue]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.5 **CsmKeys**

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

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

10.2.6 **CsmKey**

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

SWS Item	**ECUC_Csm_00015 :**		
Name	CsmKeyId		
Description	Identifier of the CsmKey		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00127 :**		
Name	CsmKeyUsePort		
Description	Does the key need RTE interfaces? True: RTE interfaces used for this key False: No RTE interfaces used for this key		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

10.2.7 **CsmPrimitives**

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAEADDecrypt	0..1	Configuration of AEAD decryption primitives
CsmAEADEncrypt	0..1	Configuration of AEAD encryption primitives
CsmDecrypt	0..1	Configurations of Decryption primitives
CsmEncrypt	0..1	Configurations of Encryption primitives
CsmHash	0..1	Container for Hash Configurations
CsmMacGenerate	0..1	Configurations of MacGenerate primitives
CsmMacVerify	0..1	Configurations of MacVerify primitives
CsmRandomGenerate	0..1	Configurations of RandomGenerate primitives
CsmSecureCounter	0..1	Configurations of SecureCounter primitives
CsmSignatureGenerate	0..1	Configurations of SignatureGenerate primitives
CsmSignatureVerify	0..1	Configurations of SignatureVerify primitives

10.2.8 **CsmQueues**

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmQueue	1..*	Container for configuration of a CSM queue. The container name serves as a symbolic name for the identifier of a queue configuration. A queue has two tasks:

		1. queue jobs which cannot be processed since the underlying hardware is busy and 2. refer to channel which shall be used

10.2.9 **CsmQueue**

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

SWS Item	**ECUC_Csm_00034 :**		
Name	CsmQueueSize		
Description	Size of the CsmQueue. If jobs cannot be processed by the underlying hardware since the hardware is busy, the jobs stay in the prioritized queue. If the queue is full, the next job will be rejected.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

10.2.10 CsmHash

SWS Item	**ECUC_Csm_00021 :**
Container Name	CsmHash
Description	Container for Hash Configurations
Configuration Parameters	

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

10.2.11 CsmHashConfig

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

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

Scope / Dependency	scope: local

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

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

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

Scope / Dependency	scope: local

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

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

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

SWS Item	**ECUC_Csm_00039 :**		
Name	CsmHashProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

10.2.12 CsmMacGenerate

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

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

10.2.13 CsmMacGenerateConfig

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

SWS Item	**ECUC_Csm_00188 :**		
Name	CsmMacGenerateAlgorithmFamiliy		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_RNG	0x16	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00133 :**
Name	CsmMacGenerateAlgorithmFamilyCustom
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmMacGenerateAlgorithmFamily
Multiplicity	0..1
Type	EcucStringParamDef

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

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

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

SWS Item	**ECUC_Csm_00136 :**
Name	CsmMacGenerateAlgorithmModeCustom
Description	Name of the custom algorithm mode used for the crypto service

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

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

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

SWS Item	**ECUC_Csm_00137 :**
Name	CsmMacGenerateDataMaxLength

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

SWS Item	**ECUC_Csm_00046 :**		
Name	CsmMacGenerateProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

10.2.14 CsmMacVerify

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

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

10.2.15 CsmMacVerifyConfig

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

SWS Item	**ECUC_Csm_00051 :**		
Name	CsmMacVerifyAlgorithmFamiliy		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
	CRYPTO_ALGOFAM_SIPHASH	0x07	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

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

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

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

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

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

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

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

SWS Item	**ECUC_Csm_00054 :**		
Name	CsmMacVerifyProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

No Included Containers

10.2.16 CsmEncrypt

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

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

10.2.17 CsmEncryptConfig

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

SWS Item	**ECUC_Csm_00182 :**		
Name	CsmEncryptAlgorithmFamiliy		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_ECIES	0x1D	
	CRYPTO_ALGOFAM_RSA	0x16	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00143 :**

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

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

SWS Item	**ECUC_Csm_00153 :**
Name	CsmEncryptAlgorithmModeCustom
Description	Name of the custom algorithm mode used for the crypto service
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--

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

SWS Item	**ECUC_Csm_00144 :**		
Name	CsmEncryptAlgorithmSecondaryFamily		
Description	Determines the algorithm family used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_NOT_SET	0x00	
Default value	CRYPTO_ALGOFAM_NOT_SET		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

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

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

SWS Item	**ECUC_Csm_00061 :**		
Name	CsmEncryptProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00147 :**		
Name	CsmEncryptResultMaxLength		
Description	Max size of the output cipher length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.18 CsmDecrypt

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmDecryptConfig	1	Container for configuration of a CSM decryption interface. The

		container name serves as a symbolic name for the identifier of an decryption interface.

10.2.19 CsmDecryptConfig

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

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

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

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

SWS Item	ECUC_Csm_00068 :		
Name	CsmDecryptAlgorithmMode		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_12ROUNDS	0x0d	
	CRYPTO_ALGOMODE_20ROUNDS	0x0e	
	CRYPTO_ALGOMODE_8ROUNDS	0x0c	
	CRYPTO_ALGOMODE_CBC	0x02	
	CRYPTO_ALGOMODE_CFB	0x03	
	CRYPTO_ALGOMODE_CTR	0x05	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_ECB	0x01	
	CRYPTO_ALGOMODE_OFB	0x04	
	CRYPTO_ALGOMODE_RSAES_OAEP	0x08	
	CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5	0x09	
	CRYPTO_ALGOMODE_XTS	0x06	
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

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

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

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

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

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

SWS Item	**ECUC_Csm_00069 :**		
Name	CsmDecryptProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

10.2.20 CsmAEADEncrypt

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

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

10.2.21 CsmAEADEncryptConfig

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

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

SWS Item	**ECUC_Csm_00184 :**		
Name	CsmAEADEncryptAlgorithmFamilyCustom		
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADEncryptAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00075 :**
Name	CsmAEADEncryptAlgorithmKeyLength
Description	Size of the AEAD encryption key in bytes
Multiplicity	1

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

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

SWS Item	**ECUC_Csm_00187 :**		
Name	CsmAEADEncryptAlgorithmModeCustom		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00159 :**		
Name	CsmAEADEncryptAssociatedDataMaxLength		
Description	Max size of the input associated data length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		

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

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

SWS Item	**ECUC_Csm_00158 :**		
Name	CsmAEADEncryptPlaintextMaxLength		
Description	Max size of the input plaintext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00077 :**		
Name	CsmAEADEncryptProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration	***Pre-compile time***	X	All Variants
	Link time	--	

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

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

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

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

No Included Containers

10.2.22 CsmAEADDecrypt

SWS Item	**ECUC_Csm_00027 :**

Container Name	CsmAEADDecrypt
Description	Configuration of AEAD decryption primitives
Configuration Parameters	

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

10.2.23 CsmAEADDecryptConfig

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

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

SWS Item	**ECUC_Csm_00185 :**		
Name	CsmAEADDecryptAlgorithmFamilyCustom		
Description	This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADDecryptAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

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

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

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

SWS Item	**ECUC_Csm_00186 :**		
Name	CsmAEADDecryptAlgorithmModeCustom		
Description	Name of the custom algorithm mode used for the crypto service		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

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

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

SWS Item	**ECUC_Csm_00165 :**		
Name	CsmAEADDecryptPlaintextMaxLength		
Description	Size of the output plaintext length in bytes		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00085 :**
Name	CsmAEADDecryptProcessing
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

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

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

Scope / Dependency	scope: local

No Included Containers

10.2.24 CsmSignatureGenerate

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

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

10.2.25 CsmSignatureGenerateConfig

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

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

SWS Item	**ECUC_Csm_00166 :**
Name	CsmSignatureGenerateAlgorithmFamilyCustom
Description	Name of the custom algorithm family used for the crypto service.

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

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

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

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

SWS Item	**ECUC_Csm_00167 :**		
Name	CsmSignatureGenerateAlgorithmSecondaryFamilyCustom		
Description	Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmSignatureGenerateAlgorithmSecondaryFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00169 :**

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

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

SWS Item	**ECUC_Csm_00092 :**		
Name	CsmSignatureGenerateProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00170 :**
Name	CsmSignatureGenerateResultLength
Description	Size of the output signature length in bytes
Multiplicity	1
Type	EcucIntegerParamDef

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

No Included Containers

10.2.26 CsmSignatureVerify

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

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

10.2.27 CsmSignatureVerifyConfig

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

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

Configuration Class	***Link time***	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00171 :**		
Name	CsmSignatureVerifyAlgorithmFamilyCustom		
Description	Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureVerifyAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

SWS Item	**ECUC_Csm_00174 :**
Name	CsmSignatureVerifyAlgorithmModeCustom
Description	Name of the custom algorithm mode used for the crypto service
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--
Post-Build Variant Value	false

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

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

SWS Item	**ECUC_Csm_00173 :**		
Name	CsmSignatureVerifyAlgorithmSecondaryFamilyCustom		
Description	Name of the custom secondary algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureVerifyAlgorithmFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_Csm_00176 :**		
Name	CsmSignatureVerifyCompareLength		
Description	Size of the input data length, for whichs signature shall be verified, in bytes		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

SWS Item	**ECUC_Csm_00099 :**		
Name	CsmSignatureVerifyProcessing		
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

No Included Containers

10.2.28 CsmSecureCounter

SWS Item	**ECUC_Csm_00030 :**
Container Name	CsmSecureCounter
Description	Configurations of SecureCounter primitives
Configuration Parameters	

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

10.2.29 CsmSecureCounterConfig

SWS Item	**ECUC_Csm_00101 :**
Container Name	CsmSecureCounterConfig
Description	Container for configuration of a CSM counter. The container name serves as a symbolic name for the identifier of a secure counter configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00102 :**		
Name	CsmSecureCounterQueueRef		
Description	This parameter refers to the queue used for that secure counter		
Multiplicity	1		
Type	Reference to [CsmQueue]		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.30 CsmRandomGenerate

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

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

10.2.31 CsmRandomGenerateConfig

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

SWS Item	**ECUC_Csm_00105 :**		
Name	CsmRandomGenerateAlgorithmFamiliy		
Description	Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOFAM_3DES	0x13	
	CRYPTO_ALGOFAM_AES	0x14	
	CRYPTO_ALGOFAM_BLAKE_1_256	0x0F	
	CRYPTO_ALGOFAM_BLAKE_1_512	0x10	
	CRYPTO_ALGOFAM_BLAKE_2s_256	0x11	
	CRYPTO_ALGOFAM_BLAKE_2s_512	0x12	
	CRYPTO_ALGOFAM_CHACHA	0x15	
	CRYPTO_ALGOFAM_CUSTOM	0xFF	
	CRYPTO_ALGOFAM_RIPEMD160	0x0E	
	CRYPTO_ALGOFAM_RNG	0x16	
	CRYPTO_ALGOFAM_SHA1	0x01	
	CRYPTO_ALGOFAM_SHA2_224	0x02	
	CRYPTO_ALGOFAM_SHA2_256	0x03	
	CRYPTO_ALGOFAM_SHA2_384	0x04	
	CRYPTO_ALGOFAM_SHA2_512	0x05	
	CRYPTO_ALGOFAM_SHA2_512_224	0x06	
	CRYPTO_ALGOFAM_SHA2_512_256	0x07	
	CRYPTO_ALGOFAM_SHA3_224	0x08	
	CRYPTO_ALGOFAM_SHA3_256	0x09	
	CRYPTO_ALGOFAM_SHA3_384	0x0A	
	CRYPTO_ALGOFAM_SHA3_512	0x0B	
	CRYPTO_ALGOFAM_SHA3_SHAKE128	0x0C	
	CRYPTO_ALGOFAM_SHA3_SHAKE256	0x0D	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope /	scope: local		

Dependency	

SWS Item	**ECUC_Csm_00177 :**		
Name	CsmRandomGenerateAlgorithmFamilyCustom		
Description	Name of the custom algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmRandomAlgorithmFamily		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00107 :**		
Name	CsmRandomGenerateAlgorithmMode		
Description	Determines the algorithm mode used for the crypto service		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	CRYPTO_ALGOMODE_CMAC	0x10	
	CRYPTO_ALGOMODE_CTRDRBG	0x12	
	CRYPTO_ALGOMODE_CUSTOM	0xFF	
	CRYPTO_ALGOMODE_GMAC	0x11	
	CRYPTO_ALGOMODE_HMAC	0x0f	
	CRYPTO_ALGOMODE_NOT_SET	0x00	
	CRYPTO_ALGOMODE_SIPHASH_2_4	0x17	
	CRYPTO_ALGOMODE_SIPHASH_4_8	0x18	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00180 :**
Name	CsmRandomGenerateAlgorithmModeCustom
Description	Name of the custom algorithm mode used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmRandomGenerateAlgorithmFamily.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	--
maxLength	--

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

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

SWS Item	**ECUC_Csm_00179 :**		
Name	CsmRandomGenerateAlgorithmSecondaryFamilyCustom		
Description	Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as Csm RandomAlgorithmSecondaryFamily.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00108 :**
Name	CsmRandomGenerateProcessing
Description	Determines how the interface shall be used for that primitive. 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	CSM_ASYNCHRONOUS	--	
	CSM_SYNCHRONOUS	--	
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

No Included Containers

10.2.32 CsmCallbacks

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

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

10.2.33 CsmCallback

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

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

SWS Item	**ECUC_Csm_00111 :**		
Name	CsmCallbackId		
Description	Identifier of the callback function.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

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