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 Release	4.2.2

Document Change History

Release	Changed by	Change Description
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
4.2.1	AUTOSAR Release Management	• Obsolete configuration elements removed • Error fixing and consistency improvements • Editorial changes
4.1.3	AUTOSAR Release Management	• Error fixing and consistency improvements • Editorial changes
4.1.2	AUTOSAR Release Management	• Error fixing and consistency improvements • Editorial changes • Removed chapter(s) on change documentation
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
4.0.3	AUTOSAR Administration	• Fixed issues with AUTOSAR Port Interfaces
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).
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:***
DEM / Dem	Diagnostic Event Manager
DET / Det	Default Error Tracer
CSM / Csm	Crypto Service Manager
CRY / Cry	Cryptographic library module

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 Crypto Service Manager
AUTOSAR_SRS_CryptoServiceManager.pdf

[15] Specification of Crypto Abstraction Library
AUTOSAR_SWS_CryptoAbstractionLibrary.pdf

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

3.2 Related standards and norms

[17] 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

n.a.

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 incorporate cryptographic library modules, which are implemented according to the cryptographic library requirement specification in chapter 8.4.
⌋ ()

[SWS_Csm_00506] ⌈ The CSM module shall use the interfaces of the incorporated cryptographic library modules to calculate the result of a cryptographic service.
⌋ ()
The incorporated cryptographic library modules provide the implementation of cryptographic routines, e.g. MD5, 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:
⌋ ()

[SWS_Csm_00006] ⌈ The code file structure shall contain one or more MISRA-C 2004 conform source files Csm_<xxx>.c, that contain the entire parts of the CSM code.
⌋ (SRS_BSW_00007)

[SWS_Csm_00692] ⌈ The code file structure shall contain one or more MISRA-C 2004 conform source files Cry_<xxx>.c, that contain the entire code of the incorporated cryptographic library modules.
⌋ (SRS_BSW_00007)

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 to be used by the underlying cryptographic library modules.

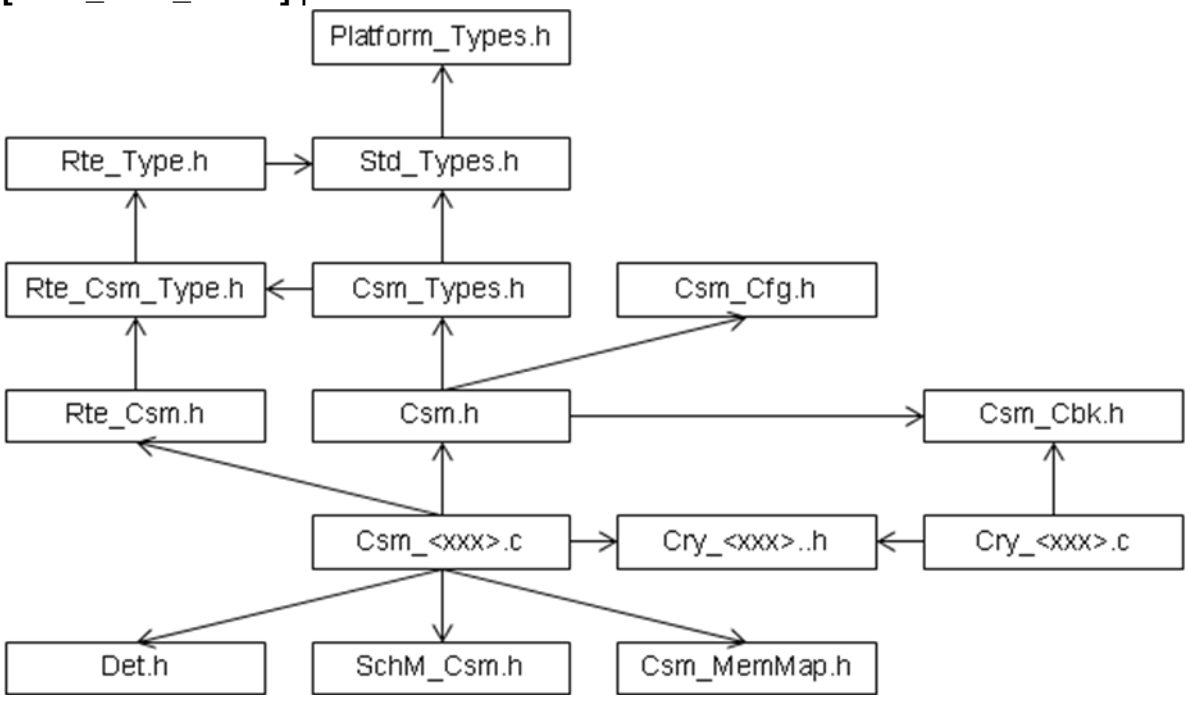
⌋ (SRS_BSW_00346)

[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_00694] ⌈ Each underlying cryptographic library module shall provide a header file Cry_<xxx>.h.
⌋ ()

[SWS_Csm_00008] ⌈ The Figure in SWS_Csm_00695 (CSM File Structure) shows the include file structure, which shall be as follows:
Csm.h shall include Csm_Types.h, Csm_Cfg.h and Csm_Cbk.h
Csm_Types.h shall include Rte_Csm_Type.h and Std_Types.h.
Csm_<xxx>.c shall include Csm.h, Rte_Csm.h, SchM_Csm.h, Csm_MemMap.h and optionally Det.h if the development error detection is turned on.
Csm_<xxx>.c shall include Cry_<xxx>.h
Cry_<xxx>.c shall include Cry_<xxx>.h and Csm_Cbk.h
⌋ (SRS_BSW_00348)

[SWS_Csm_00695] ⌈



⌋ (SRS_BSW_00348)

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_Csm_00001
-	-	SWS_Csm_00002
-	-	SWS_Csm_00004
-	-	SWS_Csm_00005
-	-	SWS_Csm_00016
-	-	SWS_Csm_00017
-	-	SWS_Csm_00019
-	-	SWS_Csm_00022
-	-	SWS_Csm_00024
-	-	SWS_Csm_00025
-	-	SWS_Csm_00026
-	-	SWS_Csm_00028
-	-	SWS_Csm_00029
-	-	SWS_Csm_00031
-	-	SWS_Csm_00035
-	-	SWS_Csm_00036
-	-	SWS_Csm_00037
-	-	SWS_Csm_00038
-	-	SWS_Csm_00039
-	-	SWS_Csm_00046
-	-	SWS_Csm_00047
-	-	SWS_Csm_00048
-	-	SWS_Csm_00050
-	-	SWS_Csm_00051
-	-	SWS_Csm_00052
-	-	SWS_Csm_00053
-	-	SWS_Csm_00054
-	-	SWS_Csm_00056
-	-	SWS_Csm_00057
-	-	SWS_Csm_00058
-	-	SWS_Csm_00059
-	-	SWS_Csm_00068
-	-	SWS_Csm_00074
-	-	SWS_Csm_00075
-	-	SWS_Csm_00079
-	-	SWS_Csm_00080

-	-	SWS_Csm_00082
-	-	SWS_Csm_00086
-	-	SWS_Csm_00087
-	-	SWS_Csm_00089
-	-	SWS_Csm_00094
-	-	SWS_Csm_00101
-	-	SWS_Csm_00108
-	-	SWS_Csm_00114
-	-	SWS_Csm_00121
-	-	SWS_Csm_00128
-	-	SWS_Csm_00134
-	-	SWS_Csm_00141
-	-	SWS_Csm_00149
-	-	SWS_Csm_00156
-	-	SWS_Csm_00163
-	-	SWS_Csm_00168
-	-	SWS_Csm_00173
-	-	SWS_Csm_00180
-	-	SWS_Csm_00187
-	-	SWS_Csm_00192
-	-	SWS_Csm_00199
-	-	SWS_Csm_00206
-	-	SWS_Csm_00212
-	-	SWS_Csm_00221
-	-	SWS_Csm_00228
-	-	SWS_Csm_00234
-	-	SWS_Csm_00243
-	-	SWS_Csm_00250
-	-	SWS_Csm_00256
-	-	SWS_Csm_00265
-	-	SWS_Csm_00272
-	-	SWS_Csm_00278
-	-	SWS_Csm_00287
-	-	SWS_Csm_00294
-	-	SWS_Csm_00300
-	-	SWS_Csm_00307
-	-	SWS_Csm_00314
-	-	SWS_Csm_00320
-	-	SWS_Csm_00327

-	-	SWS_Csm_00335
-	-	SWS_Csm_00341
-	-	SWS_Csm_00348
-	-	SWS_Csm_00355
-	-	SWS_Csm_00362
-	-	SWS_Csm_00371
-	-	SWS_Csm_00375
-	-	SWS_Csm_00377
-	-	SWS_Csm_00396
-	-	SWS_Csm_00404
-	-	SWS_Csm_00411
-	-	SWS_Csm_00418
-	-	SWS_Csm_00425
-	-	SWS_Csm_00432
-	-	SWS_Csm_00436
-	-	SWS_Csm_00443
-	-	SWS_Csm_00450
-	-	SWS_Csm_00454
-	-	SWS_Csm_00469
-	-	SWS_Csm_00478
-	-	SWS_Csm_00480
-	-	SWS_Csm_00483
-	-	SWS_Csm_00484
-	-	SWS_Csm_00485
-	-	SWS_Csm_00486
-	-	SWS_Csm_00506
-	-	SWS_Csm_00535
-	-	SWS_Csm_00537
-	-	SWS_Csm_00542
-	-	SWS_Csm_00543
-	-	SWS_Csm_00544
-	-	SWS_Csm_00657
-	-	SWS_Csm_00658
-	-	SWS_Csm_00659
-	-	SWS_Csm_00661
-	-	SWS_Csm_00662
-	-	SWS_Csm_00663
-	-	SWS_Csm_00665
-	-	SWS_Csm_00666

-	-	SWS_Csm_00667
-	-	SWS_Csm_00668
-	-	SWS_Csm_00669
-	-	SWS_Csm_00670
-	-	SWS_Csm_00671
-	-	SWS_Csm_00672
-	-	SWS_Csm_00673
-	-	SWS_Csm_00674
-	-	SWS_Csm_00675
-	-	SWS_Csm_00676
-	-	SWS_Csm_00680
-	-	SWS_Csm_00682
-	-	SWS_Csm_00684
-	-	SWS_Csm_00691
-	-	SWS_Csm_00694
-	-	SWS_Csm_00700
-	-	SWS_Csm_00701
-	-	SWS_Csm_00702
-	-	SWS_Csm_00703
-	-	SWS_Csm_00704
-	-	SWS_Csm_00728
-	-	SWS_Csm_00732
-	-	SWS_Csm_00733
-	-	SWS_Csm_00734
-	-	SWS_Csm_00736
-	-	SWS_Csm_00737
-	-	SWS_Csm_00738
-	-	SWS_Csm_00739
-	-	SWS_Csm_00740
-	-	SWS_Csm_00741
-	-	SWS_Csm_00742
-	-	SWS_Csm_00743
-	-	SWS_Csm_00744
-	-	SWS_Csm_00745
-	-	SWS_Csm_00746
-	-	SWS_Csm_00747
-	-	SWS_Csm_00748
-	-	SWS_Csm_00749
-	-	SWS_Csm_00750

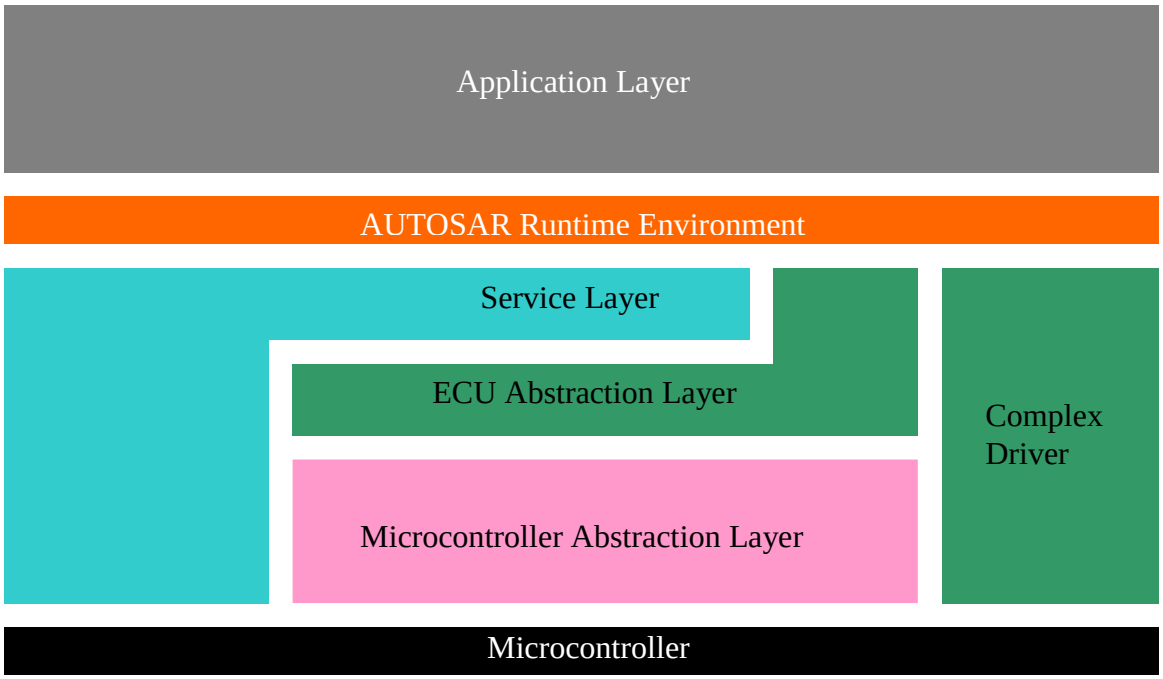
-	-	SWS_Csm_00773
-	-	SWS_Csm_00804
-	-	SWS_Csm_00806
-	-	SWS_Csm_00807
-	-	SWS_Csm_00808
-	-	SWS_Csm_00809
-	-	SWS_Csm_00810
-	-	SWS_Csm_00811
-	-	SWS_Csm_00814
-	-	SWS_Csm_00815
-	-	SWS_Csm_00816
-	-	SWS_Csm_00818
-	-	SWS_Csm_00819
-	-	SWS_Csm_00820
-	-	SWS_Csm_00821
-	-	SWS_Csm_00822
-	-	SWS_Csm_00823
-	-	SWS_Csm_00826
-	-	SWS_Csm_00827
-	-	SWS_Csm_00828
-	-	SWS_Csm_00829
-	-	SWS_Csm_00830
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2004 Standard.	SWS_Csm_00006, SWS_Csm_00692
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_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_Csm_00030
SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_Csm_00015, SWS_Csm_00488
SRS_BSW_00337	Classification of development errors	SWS_Csm_00488, 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
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Csm_00073, SWS_Csm_00455, SWS_Csm_00457
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 functions for read/receive and write/transmit data path	SWS_Csm_00479
SRS_Csm_00001	The CSM shall guarantee that the unused cryptographic primitives of the underlying crypto library are not compiled into the binary	SWS_Csm_00015

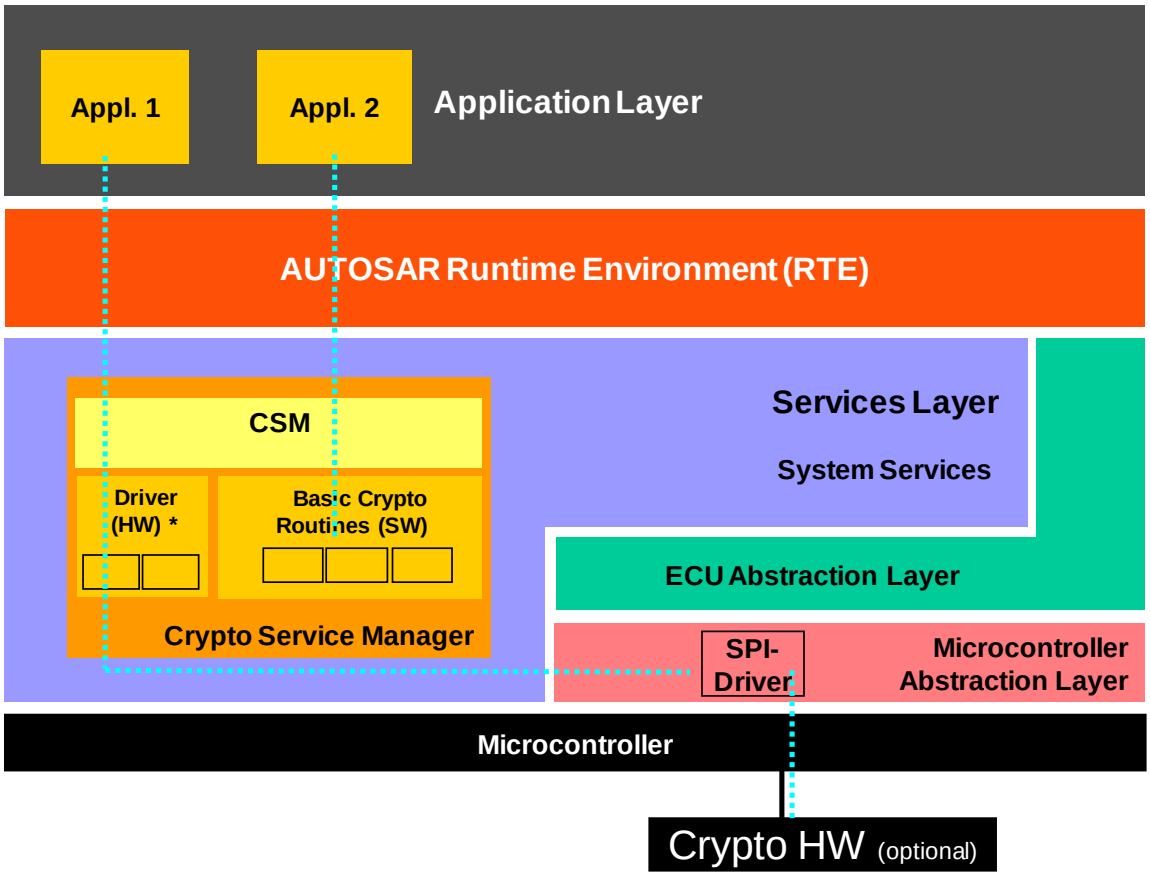
SRS_Csm_00004	The CSM shall provide configuration rules and constraints to enable plausibility checks of configuration during ECU configuration time where possible.	SWS_Csm_00030
SRS_Csm_00005	The job processing mode (synchronous or asynchronous) of the CSM shall be defined by statical configuration	SWS_Csm_00505
SRS_Csm_00006	The set of cryptographic services provided by the CSM shall be defined by statical configuration	SWS_Csm_00461
SRS_Csm_00008	The CSM module specification shall specify how the callback function has to be implemented, if the asynchronous job processing mode is selected	SWS_Csm_00032
SRS_Csm_00030	The CSM module shall use the streaming approach for most provided services	SWS_Csm_00023
SRS_Csm_00066	The CSM shall provide an interface to be accessible via the RTE	SWS_Csm_00775, SWS_Csm_00776, SWS_Csm_00777, SWS_Csm_00778, SWS_Csm_00779, 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_00788, 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_00824, SWS_Csm_00825, SWS_Csm_00831, SWS_Csm_00832, SWS_Csm_00833, SWS_Csm_00834, SWS_Csm_00835, SWS_Csm_00836, SWS_Csm_00837, SWS_Csm_00838, SWS_Csm_00839, SWS_Csm_00840, SWS_Csm_00841, SWS_Csm_00842, SWS_Csm_00843, SWS_Csm_00844, 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_00900, 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

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 software library (above: “basic crypto routines”) or on a hardware module (above: “Crypto HW”). Also, mixed setups are possible, for example if the hardware module cannot supply the necessary functionality on its own. In the following, we refer to all instantiations of underlying functionality, be it hardware or software, as “crypto library”.

Note that hardware modules may include secure key storage capability. This means that secrets are protected by hardware means and can thus not be read or altered by malicious software. To be able to support this feature, the opaque key types shall be interpreted as key handles instead of raw key data. Furthermore, parameters that are used for key output are INOUT to allow the specification of a target handle and such by the caller.

Many CRY/CPL[1] interfaces use the same cryptographic building blocks. Thus, cryptographic building blocks should be implemented as separate modules and be called from the CRY/CPL interfaces. This implies that the code for cryptographic building blocks should not be implemented more than once.

[SWS_Csm_00015] ⌈ Due to memory restrictions the CSM module and the underlying Crypto Library shall only provide those services and algorithms which are necessary for the applications running on the ECU. Therefore parts of the CSM module have to be generated based on a configuration that describes which cryptographic methods are necessary for the applications.
⌋ (SRS_BSW_00171, SRS_Csm_00001)

7.2 General behavior

[SWS_Csm_00016] ⌈ The CSM module shall only support processing of a single instance of each service at a time.
⌋ ()

[1] CPL is defined by the Crypto Abstraction Library (see [15])

[SWS_Csm_00022] ⌈ The CSM module shall allow parallel access to different services.
⌋ ()

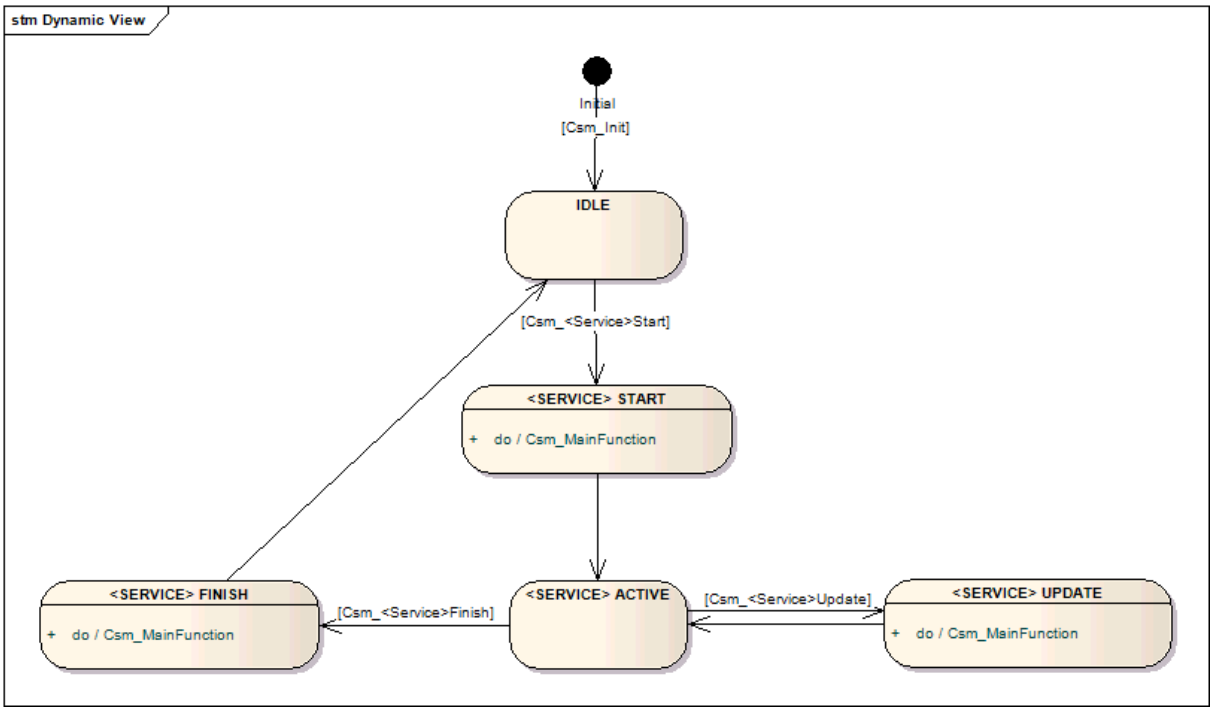
[SWS_Csm_00017] ⌈ If a service of the CSM module is requested and this service is not idle (processing currently performed), the CSM module shall reject the service request by letting the interface function return the value CSM_E_BUSY.
⌋ ()

[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 the processing of the services via a state machine.
⌋ ()

7.2.1 Normal operation

[SWS_Csm_00023] ⌈ The implementation of those CSM services which expect arbitrary amounts of user data (i.e. the hashing or encryption service) shall be based on the streaming approach with start, update and finish functions. The diagram in SWS_Csm_0024 shows the general design of such a CSM service.
⌋ (SRS_Csm_00030)

[SWS_Csm_00024] ⌈



⌋ ()

7.2.2 Functional requirements

7.2.2.1 Configuration

[SWS_Csm_00025] ⌈ Each service configuration shall be realized as a constant structure of type Csm_<Service>ConfigType .
⌋ ()

[SWS_Csm_00026] ⌈ Each service configuration shall have a name which can be configured.
⌋ ()

[SWS_Csm_00028] ⌈ It shall be possible to create several service configurations for each cryptographic service. The maximum amount of configurations is limited (see chapter 10).
⌋ ()

[SWS_Csm_00029] ⌈ When creating a service configuration, it shall be possible to configure all available and allowed schemes and underlying cryptographic primitives
⌋ ()

[SWS_Csm_00030] ⌈ It shall be checked during configuration that only valid service configurations are chosen.
⌋ (SRS_BSW_00167, SRS_Csm_00004)

[SWS_Csm_00031] ⌈ It shall be possible to configure synchronous or asynchronous job processing.
⌋ ()

[SWS_Csm_00032] ⌈ If the asynchronous interface is chosen, each service configuration shall contain a callback function.
⌋ (SRS_Csm_00008)

7.2.2.2 Synchronous job processing

[SWS_Csm_00035] ⌈ When the synchronous interface is used, the interface functions shall immediately compute the result.
⌋ ()

7.2.2.3 Asynchronous job processing

[SWS_Csm_00036] ⌈ If the asynchronous interface is used, the interface functions shall only hand over the necessary information to the service. The actual computation shall be done by the main function.
⌋ ()

[SWS_Csm_00037] ⌈ For each asynchronous request a notification of the caller after completion of the job shall be a configurable option. A callback interface has to be provided by the CSM module.
⌋ ()

[SWS_Csm_00038] ⌈ The Csm_MainFunction shall perform the processing of the services of the CSM module.
⌋ ()

[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 service configuration.
⌋ ()

7.2.3 Design notes

7.2.3.1 CSM module startup

The Csm_Init request shall not be responsible to trigger the initialization of the underlying crypto library. It is assumed, that the unerlaying crypto library 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.

7.2.3.2 Synchronisation between application and CSM module

[SWS_Csm_00734] ⌈ CSM services, which do not expect arbitrary amounts of user data, only have to provide an API Csm_<Service>() (e.g. Csm_RandomGenerate).
⌋ ()

These services shall be handled as simple function calls.

CSM services, which expect arbitrary amounts of user data, shall provide the APIs Csm_<Service>Start(), Csm_<Service>Update() and Csm_<Service>Finish(). To minimize locking and unlocking overhead or the use of other synchronization methods for CSM services which expect arbitrary amounts of user data, the communication between applications and these CSM services shall follow a strict sequence of steps which is described below. This ensures a reliable communication between applications and the CSM module.

All applications have to keep with the following rules:

7.2.3.3 Initialization

[SWS_Csm_00046] ⌈ The application calls the Csm_<Service>Start request, passing a valid service configuration to the start function. The start function shall check the validity of the configuration it receives.
⌋ ()

[SWS_Csm_00537] ⌈ If an instance of this service is being processed when Csm_<Service>Start is called, the function shall return CSM_E_BUSY.
⌋ ()

[SWS_Csm_00047] ⌈ If the synchronous interface is used, and no instance of this service is being processed when Csm_<Service>Start is called, the function shall configure the CSM immediately, set the status of the current service to active, and return the status of the service.
⌋ ()

[SWS_Csm_00048] ⌈ If the asynchronous interface is used, and no instance of this service is being processed when Csm_<Service>Start is called, the function shall return with E_OK. The main function shall process the actual initialization and set the status of the current service to active. After completing the initialization, the main function shall call the callback function as given in the service configuration.
⌋ ()

7.2.3.4 Update

The application provides the data necessary for the computation of the intended service.

[SWS_Csm_00050] ⌈ The application calls the Csm_<Service>Update request, passing data which is necessary for the computation of the service to the update function. The update function shall check whether the current service is already initialized.
⌋ ()

[SWS_Csm_00657] ⌈ If the service has not been initialized before, the update function shall return with E_NOT_OK.
⌋ ()

[SWS_Csm_00051] ⌈ The CSM shall assume that the data provided to Csm_<Service>Update will not change until it returns in case of synchronous processing, or until the callback function is called in case of asynchronous processing.
⌋ ()

[SWS_Csm_00052] ⌈ If the synchronous interface is used, and the service has been initialized before, the update function shall immediately process the given data, set the status of the current service again to active, and return the status of the update.
⌋ ()

[SWS_Csm_00053] ⌈ If the asynchronous interface is used, and the service has been initialized before, the update function shall return with E_OK. The main function shall process the actual data update and set the status of the service again to active. After completing the update the main function shall call the callback function as given

in the service configuration.
⌋ ()

[SWS_Csm_00054] ⌈ The CSM shall allow the application to call the update function arbitrarily often.
⌋ ()

7.2.3.5 Finish

The application provides the result buffer necessary for the finishing of the computation of the intended service.

[SWS_Csm_00056] ⌈ The application calls the Csm_<Service>Finish request, passing the result buffer and optional data which is necessary for the finishing of the cryptographic service to the finish function. The finish function shall check whether the current service is already initialized.
⌋ ()

[SWS_Csm_00658] ⌈ If,the service has not been initialized before, the finish function shall return with E_NOT_OK.
⌋ ()

[SWS_Csm_00057] ⌈ The CSM shall assume that the data provided to Csm_<Service>Finish will not change until it returns in case of synchronous processing, or until the callback function is called in case of asynchronous processing.
⌋ ()

[SWS_Csm_00058] ⌈ If the synchronous interface is used, and the service has been initialized before, the finish function shall immediately process the given data, finish the computation of the current cryptographic service, store the result of the service in the result buffer, set the status of the service to idle, and return the status of the finishing.
⌋ ()

[SWS_Csm_00059] ⌈ If the asynchronous interface is used, and the service has been initialized before, the finish function shall return with E_OK. The main function shall process the actual result computation and storage, and set the status of the service to idle. After completing this computation the main function shall call the callback function as given in the service configuration.
⌋ ()

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 initialisation of CSM module	CSM_E_UNINIT	0x05
Initialization of CSM module failed	CSM_E_INIT_FAILED	0x07

] ()

7.3.2 Runtime Errors

[SWS_Csm_00829][Runtime Error Types

Type of error	*Related error code*	*Value [hex]*
Requested service is not initialized	CSM_E_SERVICE_NOT_STARTED	0x02
API request called with invalid parameter (invalid method for selected service)	CSM_E_PARAM_METHOD_INVALID	0x03
API request called with invalid parameter (invalid key type for selected service)	CSM_E_PARAM_KEY_TYPE_INVALID	0x04
Provided buffer for storing the result of a computation is too small.	CSM_E_BUFFER_TOO_SMALL	0x06

] ()

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_00488] ⌈ If the development error detection is enabled and an error is detected, the desired service shall return with E_NOT_OK, in all cases except where the function returns CSM_E_SMALL_BUFFER.
⌋ (SRS_BSW_00171, SRS_BSW_00337)

[SWS_Csm_00830] ⌈ If the API returns CSM_E_SMALL_BUFFER, additonally CSM_E_BUFFER_TOO_SMALL shall be reported to the Det.
⌋ ()

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

[SWS_Csm_00539] ⌈

API call	***Error condition***	***DET related error value***
Csm_<Service>Start	CSM is not initialized	CSM_E_UNINIT
Csm_MainFunction		
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.
Csm_<Service>Update	Service is not initialized	CSM_E_SERVICE_NOT_STARTED
Csm_<Service>Finish	Service is not initialized	CSM_E_SERVICE_NOT_STARTED
Csm_<Service>Start	Invalid cryptographic method for selected service	CSM_E_PARAM_METHOD_INVALID
Csm_MacGenerateStart	Invalid key type for selected service	CSM_E_PARAM_KEY_TYPE_INVALID
Csm_MacVerifyStart		
Csm_SymBlockEncryptStart		
Csm_SymBlockDecryptStart		
Csm_SymEncryptStart		
Csm_SymDecryptStart		
Csm_AsymEncryptStart		
Csm_AsymDecryptStart		
Csm_SymKeyGenerate		
Csm_KeyDeriveSymKey		
Csm_KeyExchangeCalcPubVal		

API call	*Error condition*	*DET related error value*
Csm_KeyExchangeCalcSecret Start		
Csm_SymKeyWrapSymStart		
Csm_SymKeyWrapAsymStart		
Csm_AsymPrivateKeyWrapSy mStart		
Csm_AsymPrivateKeyWrapAsy mStart		

](SRS_BSW_00406, SRS_BSW_00337, SRS_BSW_00385)

7.5 Error notification

n/a

7.6 Debugging concept

[SWS_Csm_00542] {OBSOLETE} [The states of the CSM state machine shall be available for debugging.
] ()

8 API specification

8.1 API

8.1.1 Imported types

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

8.1.2 Type definitions

8.1.3 API types

8.1.3.1 Csm_CallbackType

[SWS_Csm_00073] ⌈

Name:	`Csm_CallbackType`
Type:	`Std_FunctionPointer`
Description:	Function pointer for the callback function which will be invoked after service completion. Signature: Std_ReturnType (*Csm_CallbackType)(Std_ReturnType)

⌋ (SRS_BSW_00359, SRS_BSW_00360)

8.1.3.2 Csm_ConfigIdType

[SWS_Csm_00691] ⌈

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

⌋ ()

8.1.3.3 Csm_<Service>ConfigType

[SWS_Csm_00074] ⌈

Name:	`Csm_<Service>ConfigType`		
Type:	`Structure`		
Element:	`Csm_ConfigIdType`	`ConfigId`	The numeric identifier of a configuration.

	Csm_CallbackType	CallbackFct	A pointer to the callback function which shall be called when the configured service has finished. This Element is only available if "CsmUseSyncJobProcessing" is "OFF". (CSM0557_Conf)
	Std_ReturnType	*PrimitiveStartFct(<primitive parameter list>)	This element shall only exist if the service contains the function Csm_<Service>Start. It is a pointer to the function Cry_<Primitive>Start of the configured cryptographic primitive. For the "primitive parameter list" see the description of Cry_<Primitive>Start.
	Std_ReturnType	*PrimitiveUpdateFct (<primitive parameter list>)	This element shall only exist if the service contains the function Csm_<Service>Update. It is a pointer to the function Cry_<Primitive>Update of the configured cryptographic primitive. For the "primitive parameter list" see the description of Cry_<Primitive>Update.
	Std_ReturnType	*PrimitiveFinishFct(<primitive parameter list>)	This element shall only exist if the service contains the function Csm_<Service>Finish. It is a pointer to the function Cry_<Primitive>Finish of the configured cryptographic primitive. For the "primitive parameter list" see the description of Cry_<Primitive>Finish.
	Std_ReturnType	*PrimitiveFct (<primitive parameter list>)	This element shall only exist if the service contains the function Csm_<Service>. It is a pointer to the function Cry_<Primitive> of the configured cryptographic primitive. For the "primitive parameter list" see the description of Cry_<Primitive>.
	void	*PrimitiveMainFct (void)	a pointer to the function Cry_<Primitive>MainFunction of the configured cryptographic primitive.
	void	*PrimitiveConfigPtr	a pointer to the configuration of the underlying cryptographic primitive
Description:	Data structure which shall encompass all information needed to specify the cryptographic primitives needed for the <Service> cryptographic service. It shall furthermore contain		

	information on the callback function.

] ()

8.1.3.4 Csm_AlignType

[SWS_Csm_00728] [

Name:	Csm_AlignType
Type:	<maxAlignScalarType>
Description:	A scalar type which has maximum alignment restrictions on the given platform. This value is configured by "CsmMaxAlignScalarType". <maxAlignScalarType> can be e.g. uint8, uint16 or uint32. All context buffers shall be aligned according to the maximum alignment of all scalar types on the given platform.

] ()

8.1.3.5 Csm_VerifyResultType

[SWS_Csm_00075] [

Name:	Csm_VerifyResultType		
Type:	uint8		
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".
Description:	Enumeration of the result type of verification operations.		

] ()

8.1.3.6 Csm_AsymPublicKeyType

[SWS_Csm_00079] [

Name:	Csm_AsymPublicKeyType		
Type:	Structure		
Element:	uint32	length	This element contains the length of the key stored in element 'data'
	Csm_AlignType [CSM_ASYM_PUB_KEY_MAX_SIZE]	data	This element contains the key data or a key handle.
Description:	Structure for the public asymmetrical key. CSM_ASYM_PUB_KEY_MAX_SIZE shall be chosen such that "CSM_ASYM_PUB_KEY_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmAsymEncryptMaxKeySize, CsmSignatureVerifyMaxKeySize, CsmAsymPublicKeyExtractMaxKeySize, CsmSymKeyWrapAsymMaxPubKeySize, CsmAsymPrivateKeyWrapAsymPubKeySize and CsmAsymPublicKeyUpdateMaxKeySize.		

] ()

8.1.3.7 Csm_AsymPrivateKeyType

[SWS_Csm_00080] [

Name:	Csm_AsymPrivateKeyType		
Type:	Structure		
Element:	uint32	length	This element contains the length of

			the key stored in element 'data'
	Csm_AlignType [CSM_ASYM_PRIV_KEY_MAX_SIZE]	data	This element contains the key data or a key handle.
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.		

] ()

8.1.3.8 Csm_SymKeyType

[SWS_Csm_00082] [

Name:	Csm_SymKeyType		
Type:	Structure		
Element:	uint32	length	This element contains the length of the key stored in element 'data'
	Csm_AlignType [CSM_SYM_KEY_MAX_SIZE]	data	This element contains the key data or a key handle.
Description:	Structure for the symmetrical key. CSM_SYM_KEY_MAX_SIZE shall be chosen such that "CSM_SYM_KEY_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmSymBlockEncryptMaxKeySize, CsmSymBlockDecryptMaxKeySize, CsmSymEncryptMaxKeySize, CsmSymDecryptMaxKeySize, CsmKeyDeriveMaxKeySize, CsmSymKeyExtractMaxKeySize, CsmMacGenerateMaxKeySize, CsmMacVerifyMaxKeySize, CsmSymKeyWrapSymMaxSymKeySize, CsmSymKeyWrapAsymMaxSymKeySize, CsmAsymPrivateKeyWrapSymMaxSymKeySize, CsmKeyExchangeCalcSymKeyMaxSymKeySize, CsmKeyDeriveSymKeyMaxSymKeySize, CsmSymKeyUpdateMaxKeySize and CsmSymKeyGenerateMaxKeySize.		

] ()

8.1.3.9 Csm_KeyExchangeBaseType

[SWS_Csm_00086] [

Name:	Csm_KeyExchangeBaseType		
Type:	Structure		
Element:	uint32	length	This element contains the length of the key stored in element 'data'
	Csm_AlignType [CSM_KEY_EX_BASE_MAX_SIZE]	data	This element contains the key data or a key handle.
Description:	Structure with base type information of the key exchange protocol. CSM_KEY_EX_BASE_MAX_SIZE shall be chosen such that "CSM_KEY_EX_BASE_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmKeyExchangeCalcPubValMaxBaseTypeSize, CsmKeyExchangeCalcSecretMaxBaseTypeSize and CsmKeyExchangeCalcSymKeyMaxBaseTypeSize.		

] ()

8.1.3.10 Csm_KeyExchangePrivateType

[SWS_Csm_00087] ⌈

Name:	Csm_KeyExchangePrivateType		
Type:	Structure		
Element:	uint32	length	This element contains the length of the key stored in element 'data'
	Csm_AlignType [CSM_KEY_EX_PRIV_MAX_SIZE]	data	This element contains the key data or a key handle.
Description:	Structure with the private Information of the key exchange protocol only known to the current user. CSM_KEY_EX_PRIV_MAX_SIZE shall be chosen such that "CSM_KEY_EX_PRIV_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmKeyExchangeCalcPubValMaxPrivateTypeSize, CsmKeyExchangeCalcSecretMaxPrivateTypeSize CsmKeyExchangeCalcSymKeyMaxPrivateTypeSize.		

⌋ ()

8.1.4 API functions

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

8.1.5 General interfaces

8.1.5.1 Csm_Init

[SWS_Csm_00646] ⌈

Service name:	Csm_Init
Syntax:	void Csm_Init(void)
Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This function shall be used to initialize 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.
⌋ ()

8.1.5.2 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)

8.1.6 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.

8.1.6.1 Csm_HashStart

[SWS_Csm_00089] ⌈

Service name:	Csm_HashStart	

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 interface shall be used to initialize the hash 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"	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_HashStart.

8.1.6.2 Csm_HashUpdate

[SWS_Csm_00094] ⌈

Service name:	Csm_HashUpdate	

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 interface shall be used to feed the hash 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 hash computation is done by the underlying primitive.	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_HashUpdate.

8.1.6.3 Csm_HashFinish

[SWS_Csm_00101] ⌈

Service name:	Csm_HashFinish	

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 interface shall be used to finish the hash service of the CSM module. 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 hash computation is done by the underlying primitive.	

] ()

[SWS_Csm_00661] [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 and truncation is allowed, the result of the computation shall be truncated to the size of the provided buffer, and `E_OK` shall be returned. If the provided buffer is too small, and truncation is not allowed, `CSM_E_SMALL_BUFFER` shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_HashFinish.

8.1.7 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.

8.1.7.1 Csm_MacGenerateStart

[SWS_Csm_00108] [

Service name:	Csm_MacGenerateStart

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 interface shall be used to initialize the MAC generate 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_MacGenerateStart.

8.1.7.2 Csm_MacGenerateUpdate

[SWS_Csm_00114] [

Service name:	Csm_MacGenerateUpdate	

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 interface shall be used to feed the MAC generate 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 MAC computation is done by the underlying primitive.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_MacGenerateUpdate.

8.1.7.3 Csm_MacGenerateFinish

[SWS_Csm_00121] [

Service name:	Csm_MacGenerateFinish	

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 interface shall be used to finish the MAC generation 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 MAC computation is done by the underlying primitive.	

] ()

[SWS_Csm_00662] [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 and truncation is allowed, the result of the computation shall be truncated to the size of the provided buffer, and `E_OK` shall be returned. If the provided buffer is too small, and truncation

is not allowed, `CSM_E_SMALL_BUFFER` shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_MacGenerateFinish.

8.1.7.4 Csm_MacVerifyStart

[SWS_Csm_00128] [

Service name:	Csm_MacVerifyStart	

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 interface shall be used to initialize the MAC verify 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_MacVerifyStart.

8.1.7.5 Csm_MacVerifyUpdate

[SWS_Csm_00134] [

Service name:	Csm_MacVerifyUpdate	

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 interface shall be used to feed the MAC verification 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 MAC computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_MacVerifyUpdate.

8.1.7.6 Csm_MacVerifyFinish

[SWS_Csm_00141] [

Service name:	Csm_MacVerifyFinish	

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 interface shall be used to finish the MAC verification 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 MAC computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_MacVerifyFinish.

8.1.8 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.

8.1.8.1 Csm_RandomSeedStart

[SWS_Csm_00149] ⌈

Service name:	Csm_RandomSeedStart	

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 interface shall be used to initialize the random seed 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".	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_RandomSeedStart.

8.1.8.2 Csm_RandomSeedUpdate

[SWS_Csm_00156] ⌈

Service name:	Csm_RandomSeedUpdate

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 interface shall be used to feed a seed to the random seed 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>Update of the primitive which is identified by the stored configuration information and return the value returned by that function. The seeding of the random number generator is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_RandomSeedUpdate.

8.1.8.3 Csm_RandomSeedFinish

[SWS_Csm_00163] [

Service name:	Csm_RandomSeedFinish	

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 interface shall be used to finish the random seed 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 seeding of the random number generator is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_RandomSeedFinish.

8.1.8.4 Csm_RandomGenerate

[SWS_Csm_00543] ⌈

Service name:	Csm_RandomGenerate	

Syntax:	`Std_ReturnType Csm_RandomGenerate(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* resultPtr,`
	`  uint32 resultLength`
	`)`	

Service ID[hex]:	0x0f	
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 random number generation.
	resultLength	holds the amount of random bytes which should be generated.
Parameters (inout):	None	
Parameters (out):	resultPtr	holds a pointer to the memory location which will hold the result of the random number generation. The memory location must have at least the size “resultLength”.
Return value:	Std_ReturnType	E_OK: request successful E_NOT_OK: request failed CSM_E_BUSY: request failed, service is still busy CSM_E_ENTROPY_EXHAUSTION: request failed, entropy of random number generator is exhausted.
Description:	This interface shall be used to start the random number generation 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> of the primitive which is identified by the "cfgId" and return the value returned by that function. If Cry_<Primitive> returned successfully, the service state has to be set to "active". The random number generation is done by the underlying primitive.	

⌋ ()

The generation of a random number is based on the seed, which was previously set with the interfaces Csm_RandomSeedStart, Csm_RandomSeedUpdate, and Csm_RandomSeedFinish. These interfaces follow the streaming approach. Thus it is possible to feed the seed e.g. from different sources.
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 requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_RandomGenerate.

8.1.9 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.

8.1.9.1 Csm_SymBlockEncryptStart

[SWS_Csm_00168] ⌈

Service name:	Csm_SymBlockEncryptStart	

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 interface shall be used to initialize the symmetrical block 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".	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymBlockEncryptStart.

8.1.9.2 Csm_SymBlockEncryptUpdate

[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 interface shall be used to feed the symmetrical block 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.	

] ()

[SWS_Csm_00663] [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, `CSM_E_SMALL_BUFFER` shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymBlockEncryptUpdate.

8.1.9.3 Csm_SymBlockEncryptFinish

[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 interface shall be used to finish the symmetrical block 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.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymBlockEncryptFinish.

8.1.9.4 Csm_SymBlockDecryptStart

[SWS_Csm_00187] [

Service name:	Csm_SymBlockDecryptStart	

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 interface shall be used to initialize the symmetrical block decrypt 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymBlockDecryptStart.

8.1.9.5 Csm_SymBlockDecryptUpdate

[SWS_Csm_00192] [

Service name:	Csm_SymBlockDecryptUpdate

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 interface shall be used to feed the symmetrical block decryption 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 decryption process is done by the underlying primitive.	

] ()

[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, `CSM_E_SMALL_BUFFER` shall be returned.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymBlockDecryptUpdate.

8.1.9.6 Csm_SymBlockDecryptFinish

[SWS_Csm_00199] [

Service name:	Csm_SymBlockDecryptFinish	
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 interface shall be used to finish the symmetrical block decryption 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 decryption process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymBlockDecryptFinish.

8.1.10 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.

8.1.10.1 Csm_SymEncryptStart

[SWS_Csm_00206] [

Service name:	Csm_SymEncryptStart	

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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymEncryptStart.

8.1.10.2 Csm_SymEncryptUpdate

[SWS_Csm_00212] [

Service name:	Csm_SymEncryptUpdate	

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.	

] ()

[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,

CSM_E_SMALL_BUFFER shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymEncryptUpdate.

8.1.10.3 Csm_SymEncryptFinish

[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, CSM_E_SMALL_BUFFER shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymEncryptFinish.

8.1.10.4 Csm_SymDecryptStart

[SWS_Csm_00228] [

Service name:	Csm_SymDecryptStart	
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 interface shall be used to initialize the symmetrical decryption 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymDecryptStart.

8.1.10.5 Csm_SymDecryptUpdate

[SWS_Csm_00234] [

Service name:	Csm_SymDecryptUpdate	
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 interface shall be used to feed the symmetrical decryption 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 decryption process is done by the underlying primitive.	

] ()

[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, `CSM_E_SMALL_BUFFER` shall be returned.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymDecryptUpdate.

8.1.10.6 Csm_SymDecryptFinish

[SWS_Csm_00243] [

Service name:	Csm_SymDecryptFinish	

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 interface shall be used to finish the symmetrical decryption 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 decryption process is done by the underlying primitive.	

⌋ ()

[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, `CSM_E_SMALL_BUFFER` shall be returned.

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymDecryptFinish.

8.1.11 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.

8.1.11.1 Csm_AsymEncryptStart

[SWS_Csm_00250] ⌈

Service name:	Csm_AsymEncryptStart	

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 interface shall be used to initialize the asymmetrical encryption 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymEncryptStart.

8.1.11.2 Csm_AsymEncryptUpdate

[SWS_Csm_00256] [

Service name:	Csm_AsymEncryptUpdate	

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 interface shall be used to feed the asymmetrical 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.	

] ()

[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,

CSM_E_SMALL_BUFFER shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymEncryptUpdate.

8.1.11.3 Csm_AsymEncryptFinish

[SWS_Csm_00265] [

Service name:	Csm_AsymEncryptFinish	
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 interface shall be used to finish the asymmetrical 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_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, CSM_E_SMALL_BUFFER shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymEncryptFinish.

8.1.11.4 Csm_AsymDecryptStart

[SWS_Csm_00272] [

Service name:	Csm_AsymDecryptStart	

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 interface shall be used to initialize the asymmetrical decryption 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymDecryptStart.

8.1.11.5 Csm_AsymDecryptUpdate

[SWS_Csm_00278] [

Service name:	Csm_AsymDecryptUpdate	

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 interface shall be used to feed the asymmetrical decryption 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 decryption process is done by the underlying primitive.	

] ()

[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, `CSM_E_SMALL_BUFFER` shall be returned.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymDecryptUpdate.

8.1.11.6 Csm_AsymDecryptFinish

[SWS_Csm_00287] [

Service name:	Csm_AsymDecryptFinish	

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 interface shall be used to finish the asymmetrical decryption 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 decryption process is done by the underlying primitive.

] ()

[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, `CSM_E_SMALL_BUFFER` shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymDecryptFinish.

8.1.12 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.

8.1.12.1 Csm_SignatureGenerateStart

[SWS_Csm_00294] [

Service name:	Csm_SignatureGenerateStart	

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 interface shall be used to initialize the signature generate 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SignatureGenerateStart.

8.1.12.2 Csm_SignatureGenerateUpdate

[SWS_Csm_00300] [

Service name:	Csm_SignatureGenerateUpdate	

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 interface shall be used to feed the signature generation 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 signature computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SignatureGenerateUpdate.

8.1.12.3 Csm_SignatureGenerateFinish

[SWS_Csm_00307] [

Service name:	Csm_SignatureGenerateFinish

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 interface shall be used to finish the signature generation 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 signature computation is done by the underlying primitive.	

] ()

[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, `CSM_E_SMALL_BUFFER` shall be returned.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SignatureGenerateFinish.

8.1.12.4 Csm_SignatureVerifyStart

[SWS_Csm_00314] [

Service name:	Csm_SignatureVerifyStart	

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 interface shall be used to initialize the signature verify 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SignatureVerifyStart.

8.1.12.5 Csm_SignatureVerifyUpdate

[SWS_Csm_00320] [

Service name:	Csm_SignatureVerifyUpdate	

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 interface shall be used to feed the signature verification 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 signature computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SignatureVerifyUpdate.

8.1.12.6 Csm_SignatureVerifyFinish

[SWS_Csm_00327] [

Service name:	Csm_SignatureVerifyFinish	

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 interface shall be used to finish the signature verification 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 signature computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SignatureVerifyFinish.

8.1.13 Compression / Decompression interface

Due to usage of compression/decompression algorithms it is possible to reduce of the amount of data, which must be processed by encryption/decryption. Due to appropriate seletion of the compression/decompression algorithm, the aggregated load can be reduced: the compression and encryption of the reduced amount of data respectively decription and decompression consumes fewer resources than the encryption and decryption of the uncompressed data.

The following APIs can be used for compression and decompression of data.

8.1.13.1 Csm_CompressStart

[SWS_Csm_00818] [

Service name:	Csm_CompressStart

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

Service ID[hex]:	0x4d
Sync/Async:	Synchronous or Async, dependent on configuration (ECUC_Csm_00557)

Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the compression 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 compression 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_CompressStart.

8.1.13.2 Csm_CompressUpdate

[SWS_Csm_00819] [

Service name:	Csm_CompressUpdate	

Syntax:	`Std_ReturnType Csm_CompressUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* plainTextPtr,`
	`  uint32 plainTextLength,`
	`  uint8* compressedTextPtr,`
	`  uint32* compressedTextLengthPtr`
	`)`	

Service ID[hex]:	0x4e	
Sync/Async:	Synchronous or Async, dependent on configuration (ECUC_Csm_00557)	
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 compressed.
	plainTextLength	contains the length of the plain text in bytes
Parameters (inout):	compressedTextLengthPtr	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 compressedTextPtr. When the request has finished, the amount of data that has been compressed shall be stored.
Parameters (out):	compressedTextPtr	holds a pointer to the memory location which will hold the compressed 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 compress 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 compression process is done by the underlying primitive.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_CompressUpdate.

8.1.13.3 Csm_CompressFinish

[SWS_Csm_00820] [

Service name:	Csm_CompressFinish	

Syntax:	`Std_ReturnType Csm_CompressFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* compressedTextPtr,`
	`  uint32* compressTextLengthPtr`
	`)`	

Service ID[hex]:	0x4f	
Sync/Async:	Synchronous or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	compressTextLengthPtr	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 compressedTextPtr. When the request has finished, the amount of data that has been compressed shall be stored.
Parameters (out):	compressedTextPtr	holds a pointer to the memory location which will hold the compressed 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 compression 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 compression process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_CompressFinish.

8.1.13.4 Csm_DecompressStart

[SWS_Csm_00821] [

Service name:	Csm_DecompressStart

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

Service ID[hex]:	0x50	
Sync/Async:	Synchronous or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the decompression 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 decompression 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_DecompressStart.

8.1.13.5 Csm_DecompressUpdate

[SWS_Csm_00822] [

Service name:	Csm_DecompressUpdate	

Syntax:	`Std_ReturnType Csm_DecompressUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* plainTextPtr,`
	`  uint32 plainTextLength,`
	`  uint8* decompressedTextPtr,`
	`  uint32* decompressedTextLengthPtr`
	`)`	

Service ID[hex]:	0x51	
Sync/Async:	Synchronous or Async, dependent on configuration (ECUC_Csm_00557)	
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 decompressed.
	plainTextLength	contains the length of the plain text in bytes
Parameters (inout):	decompressedTextLengthPtr	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 decompressedTextPtr. When the request has finished, the amount of data that has been decompressed shall be stored.
Parameters (out):	decompressedTextPtr	holds a pointer to the memory location which will

		hold the decompressed 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 decompress 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 decompression process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_DecompressUpdate.

8.1.13.6 Csm_DecompressFinish

[SWS_Csm_00823] ⌈

Service name:	Csm_DecompressFinish	

Syntax:	`Std_ReturnType Csm_DecompressFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* decompressTextPtr,`
	`  uint32* decompressTextLengthPtr`
	`)`	

Service ID[hex]:	0x52	
Sync/Async:	Synchronous or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	Holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	decompressTextLengthPtr	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 decompressedTextPtr. When the request has finished, the amount of data that has been decompressed shall be stored.
Parameters (out):	decompressTextPtr	holds a pointer to the memory location which will hold the decompressed 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 decompression 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 decompression process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_DecompressFinish.

8.1.14 Checksum interface

The goal of checksum algorithms is to detect accidental modification such as corruption to stored data or errors in a communication channel. They are not designed to detect intentional corruption by a malicious agent. Indeed, many checksum algorithms can be easily inverted, in the sense that one can easily modify the data so as to preserve its checksum.

8.1.14.1 Csm_ChecksumStart

[SWS_Csm_00335] ⌈

Service name:	Csm_ChecksumStart	

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 interface shall be used to initialize the checksum 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".	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_ChecksumStart.

8.1.14.2 Csm_ChecksumUpdate

[SWS_Csm_00341] ⌈

Service name:	Csm_ChecksumUpdate

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 interface shall be used to feed the checksum 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 checksum computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_ChecksumUpdate.

8.1.14.3 Csm_ChecksumFinish

[SWS_Csm_00348] [

Service name:	Csm_ChecksumFinish	

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 interface shall be used to finish the checksum 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 checksum computation is done by the underlying primitive.	

] ()

[SWS_Csm_00674] [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 and truncation is allowed, the result of the computation shall be truncated to the size of the provided buffer, and `E_OK` shall be returned. If the provided buffer is too small, and truncation is not allowed, `CSM_E_SMALL_BUFFER` shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_ChecksumFinish.

8.1.15 Key generation interface

For key generation the following interfaces can be used.

Note that these interfaces may be used for key transport purposes. In this case, any necessary auxiliary information (e.g., wrapping key, shared information, randomness) will have to be encoded unambiguously into the data provided in the dataPtr buffer.

8.1.15.1 Csm_SymKeyGenerate

[SWS_Csm_00804] [

Service name:	Csm_SymKeyGenerate	

Syntax:	`Std_ReturnType Csm_SymKeyGenerate(`
	`  Csm_ConfigIdType cfgId,`
	`  Csm_SymKeyType* keyPtr,`
	`  uint32 keyLength`
	`)`	

Service ID[hex]:	0x53	
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 generation.
	keyLength	the requested length of the symmetric key counted in bytes. Note: any template may have already contained this information. However, there are cases where the application should not be required to understand the semantics of the Csm_SymKeyType struct. Having an extra parameter allows the specification without this knowledge.
Parameters (inout):	keyPtr	is a pointer to the target key buffer that may already contain a template for the symmetric key.

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 generate a symmetric key. 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> of the primitive which is identified by the "cfgId" and return the value returned by that function. If Cry_<Primitive> returned successfully, the service state has to be set to "active". The key generation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyGenerate.

8.1.16 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_KeyDeriveSymKey interface.

8.1.16.1 Csm_KeyDeriveStart

[SWS_Csm_00355] [

Service name:	Csm_KeyDeriveStart	

Syntax:	`Std_ReturnType Csm_KeyDeriveStart(`
	`  Csm_ConfigIdType cfgId,`
	`  uint32 keyLength,`
	`  uint32 iterations`
	`)`	

Service ID[hex]:	0x2b	
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 derivation
	keyLength	holds the length of the key to be derived by the underlying key derivation primitive.
	iterations	holds the number of iterations to be performed by the underlying key derivation primitive
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 key derivation 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyDeriveStart.

8.1.16.2 Csm_KeyDeriveUpdate

[SWS_Csm_00362] ⌈

Service name:	Csm_KeyDeriveUpdate	

Syntax:	`Std_ReturnType Csm_KeyDeriveUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* passwordPtr,`
	`  uint32 passwordLength,`
	`  const uint8* saltPtr,`
	`  uint32 saltLength`
	`)`	

Service ID[hex]:	0x2c	
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.
	passwordPtr	holds a pointer to the password, i.e. the original key, from which to derive a new key.
	passwordLength	holds the length of the password in bytes
	saltPtr	holds a pointer to the cryptographic salt, i.e. a random number, for the underlying primitive
	saltLength	holds the length of the salt 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 interface shall be used to feed the key derivation 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 key derivation computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyDeriveUpdate.

8.1.16.3 Csm_KeyDeriveFinish

[SWS_Csm_00371] ⌈

Service name:	Csm_KeyDeriveFinish	

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

Service ID[hex]:	0x2d	
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 the memory location which will hold the result of the key derivation.
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 finish the key derivation 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 key derivation computation is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyDeriveFinish.

8.1.16.4 Csm_KeyDeriveSymKey

[SWS_Csm_00375] [

Service name:	Csm_KeyDeriveSymKey	

Syntax:	`Std_ReturnType Csm_KeyDeriveSymKey(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_SymKeyType* baseKeyPtr,`
	`  const uint8* customisationValPtr,`
	`  uint32 customisationValLength,`
	`  Csm_SymKeyType* derivedKeyPtr`
	`)`	

Service ID[hex]:	0x4c	
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 derivation
	baseKeyPtr	holds a pointer to the key from which the new key shall be derived
	customisationValPtr	holds a pointer to the customisation value (if any)
	customisationValLength	holds the length of the customisation value in bytes
Parameters (inout):	derivedKeyPtr	holds a pointer to the memory location which will hold the result of the key derivation.
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 key derivation from key 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> of the primitive which is identified by the "cfgId" and return the value returned by that function. If Cry_<Primitive> returned successfully, the service state has to be set to "active". The key derivation is done by the underlying primitive.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyDeriveSymKey.

8.1.17 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.
Derivation of secret keys that shall be protected by hardware means can be done by using the Csm_KeyExchangeCalcSymKey interface.

8.1.17.1 Csm_KeyExchangeCalcPubVal

[SWS_Csm_00377] [

Service name:	Csm_KeyExchangeCalcPubVal	

Syntax:	`Std_ReturnType Csm_KeyExchangeCalcPubVal(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_KeyExchangeBaseType* basePtr,`
	`  const Csm_KeyExchangePrivateType* privateValuePtr,`
	`  uint8* publicValuePtr,`
	`  uint32* publicValueLengthPtr`
	`)`	

Service ID[hex]:	0x2e	
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 key exchange
	basePtr	holds a pointer to the base information known to both users of the key exchange protocol
	privateValuePtr	holds a pointer to the private information known only to the current user of the key exchange protocol
Parameters (inout):	publicValueLengthPtr	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 publicValuePtr. When the request has finished, the actual length of the calculated public value shall be stored.
Parameters (out):	publicValuePtr	holds a pointer to the memory location which will hold the public value of the key exchange protocol
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 start the public value calculation 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> of the primitive which is identified by the "cfgId" and return the value returned by that function. If Cry_<Primitive> returned successfully, the service state has to be set to "active". The calculation of the public value is done by the underlying primitive.	

] ()

[SWS_Csm_00675] [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, `CSM_E_SMALL_BUFFER` shall be returned.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcPubVal.

8.1.17.2 Csm_KeyExchangeCalcSecretStart

[SWS_Csm_00396] [

Service name:	Csm_KeyExchangeCalcSecretStart	

Syntax:	`Std_ReturnType Csm_KeyExchangeCalcSecretStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_KeyExchangeBaseType* basePtr,`
	`  const Csm_KeyExchangePrivateType* privateValuePtr`
	`)`	

Service ID[hex]:	0x2f	
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 key exchange
	basePtr	holds a pointer to the base information known to both users of the key exchange protocol
	privateValuePtr	holds a pointer to the private information known only to the current user of the key exchange protocol
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 key exchange 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcSecretStart.

8.1.17.3 Csm_KeyExchangeCalcSecretUpdate

[SWS_Csm_00404] [

Service name:	Csm_KeyExchangeCalcSecretUpdate	
Syntax:	`Std_ReturnType Csm_KeyExchangeCalcSecretUpdate( Csm_ConfigIdType cfgId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength )`	
Service ID[hex]:	0x30	
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.
	partnerPublicValuePtr	holds a pointer to the data representing the public value of the key exchange partner
	partnerPublicValueLength	contains the length of the part of the partner value 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 interface shall be used to feed the key exchange service with the public value coming from the partner of the key exchange protocol. 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 calculation of the shared secret is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcSecretUpdate.

8.1.17.4 Csm_KeyExchangeCalcSecretFinish

[SWS_Csm_00411] [

Service name:	Csm_KeyExchangeCalcSecretFinish
Syntax:	`Std_ReturnType Csm_KeyExchangeCalcSecretFinish( Csm_ConfigIdType cfgId, uint8* sharedSecretPtr, uint32* sharedSecretLengthPtr,`

	boolean TruncationIsAllowed)	
Service ID[hex]:	0x31	
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):	sharedSecretLengthPtr	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 sharedSecretPtr. When the request has finished, the actual length of the computed value shall be stored.
Parameters (out):	sharedSecretPtr	holds a pointer to the memory location which will hold the result of the key exchange. 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 result provided buffer is too small to store the result, and truncation was not allowed.
Description:	This interface shall be used to finish the key exchange 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 calculation of the shared secret is done by the underlying primitive.	

] ()

[SWS_Csm_00676] [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 and truncation is allowed, the result of the computation shall be truncated to the size of the provided buffer, and `E_OK` shall be returned. If the provided buffer is too small, and truncation is not allowed, `CSM_E_SMALL_BUFFER` shall be returned.
] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcSecretFinish.

8.1.17.5 Csm_KeyExchangeCalcSymKeyStart

[SWS_Csm_00736] [

Service name:	Csm_KeyExchangeCalcSymKeyStart
Syntax:	Std_ReturnType Csm_KeyExchangeCalcSymKeyStart(Csm_ConfigIdType cfgId, const Csm_KeyExchangeBaseType* basePtr, const Csm_KeyExchangePrivateType* privateValuePtr)

Service ID[hex]:	0x3d	
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 key exchange
	basePtr	holds a pointer to the base information known to both users of the key exchange protocol
	privateValuePtr	holds a pointer to the private information known only to the current user of the key exchange protocol
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 key exchange 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcSymKeyStart.

8.1.17.6 Csm_KeyExchangeCalcSymKeyUpdate

[SWS_Csm_00737] [

Service name:	Csm_KeyExchangeCalcSymKeyUpdate	

Syntax:	`Std_ReturnType Csm_KeyExchangeCalcSymKeyUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  const uint8* partnerPublicValuePtr,`
	`  uint32 partnerPublicValueLength`
	`)`	

Service ID[hex]:	0x3e	
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.
	partnerPublicValuePtr	holds a pointer to the data representing the public value of the key exchange partner
	partnerPublicValueLength	contains the length of the part of the partner value 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 interface shall be used to feed the key exchange service with the public value coming from the partner of the key exchange protocol. 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 calculation of the shared secret is done by the underlying primitive.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcSymKeyUpdate.

8.1.17.7 Csm_KeyExchangeCalcSymKeyFinish

[SWS_Csm_00738] [

Service name:	Csm_KeyExchangeCalcSymKeyFinish	
Syntax:	`Std_ReturnType Csm_KeyExchangeCalcSymKeyFinish( Csm_ConfigIdType cfgId, Csm_SymKeyType* sharedKeyPtr )`	
Service ID[hex]:	0x3f	
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):	sharedKeyPtr	holds a pointer to the key which will hold the result of the key exchange
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 finish the key exchange 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 calculation of the shared secret is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_KeyExchangeCalcSymKeyFinish.

8.1.18 Symetrical key update interface

Symmetrical key update interface is used to update a symmetrical key from certain data sources.

Note that these interfaces may be used for key transport purposes. In this case, any necessary auxiliary information (e.g., wrapping key, shared information, randomness) will have to be encoded unambiguously into the data provided in the dataPtr buffer.

8.1.18.1 Csm_SymKeyUpdateStart

[SWS_Csm_00814] ⌈

Service name:	Csm_SymKeyUpdateStart	

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

Service ID[hex]:	0x54	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the Symmetric Key update
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 symmetric key update 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".	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyUpdateStart.

8.1.18.2 Csm_SymKeyUpdateUpdate

[SWS_Csm_00815] ⌈

Service name:	Csm_SymKeyUpdateUpdate	

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

Service ID[hex]:	0x55	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
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 a buffer containing one chunk of the key update data stream.
	dataLength	contains the length of the given chunk at dataPtr counted 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 interface shall be used to feed the Symmetric Key Update 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 symmetric Key Update process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyUpdateUpdate.

8.1.18.3 Csm_SymKeyUpdateFinish

[SWS_Csm_00816] [

Service name:	Csm_SymKeyUpdateFinish	

Syntax:	`Std_ReturnType Csm_SymKeyUpdateFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  Csm_SymKeyType* targetKeyPtr,`
	`  uint8* proofPtr,`
	`  uint32* proofLengthPtr`
	`)`	

Service ID[hex]:	0x56	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	targetKeyPtr	holds a pointer to the target key buffer that may already contain a template for the symmetric key.
	proofLengthPtr	holds a pointer to the length associated with the buffer at proofPtr. The caller shall fill in the size of the provided buffer and CSM shall return the number of actually used bytes in it.
Parameters (out):	proofPtr	holds a pointer to a buffer that will hold the cryptographic proof that update has been successful.
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 Symmetric Key Update 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 Symmetric Key Update process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyUpdateFinish.

8.1.19 Symmetrical key extract interface

Symmetrical key extract interface is used to extract a symmetrical key structure from certain data sources.
Note that this interface may be used for key transport purposes. In this case, any necessary auxiliary information (e.g., wrapping key, shared information, randomness) will have to be encoded unambiguously into the data provided in the dataPtr buffer.

8.1.19.1 Csm_SymKeyExtractStart

[SWS_Csm_00418] ⌈

Service name:	Csm_SymKeyExtractStart	

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 interface shall be used to initialize the symmetrical key extraction 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".	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyExtractStart.

8.1.19.2 Csm_SymKeyExtractUpdate

[SWS_Csm_00425] ⌈

Service name:	Csm_SymKeyExtractUpdate

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 interface shall be used to feed the symmetrical key extraction service with 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 calculation of the extraction algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyExtractUpdate.

8.1.19.3 Csm_SymKeyExtractFinish

[SWS_Csm_00432] [

Service name:	Csm_SymKeyExtractFinish	

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 interface shall be used to finish the symmetrical key extraction 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 calculation of the extraction algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyExtractFinish.

8.1.20 Symmetrical key wrapping interface

Symmetrical key wrapping interface is used to export a symmetrical key structure, e.g. to be used on a different device. To be able to use symmetric and asymmetric wrapping keys, two different interfaces are standardised.

8.1.20.1 Csm_SymKeyWrapSymStart

[SWS_Csm_00739] [

Service name:	Csm_SymKeyWrapSymStart	

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

Service ID[hex]:	0x40	
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 wrapping
	keyPtr	holds a pointer to the symmetric key to be wrapped
	wrappingKeyPtr	holds a pointer to the key used for wrapping
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 key wrapping 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyWrapSymStart.

8.1.20.2 Csm_SymKeyWrapSymUpdate

[SWS_Csm_00740] [

Service name:	Csm_SymKeyWrapSymUpdate
Syntax:	`Std_ReturnType Csm_SymKeyWrapSymUpdate(` `  Csm_ConfigIdType cfgId,`

	uint8* dataPtr, uint32* dataLengthPtr)	
Service ID[hex]:	0x41	
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):	dataLengthPtr	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 dataPtr. When the request has finished, the actual length of the computed value shall be stored.
Parameters (out):	dataPtr	holds a pointer to the memory location which will hold the first chunk of the result of the key wrapping. If the result does not fit into the given buffer, the caller shall call the service again, until *dataLengthPtr is equal to zero, indicating that the complete result has been retrieved.
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 retrieve the result of the key wrapping operation from the symmetrical key wrapping 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>Update of the primitive which is identified by the stored configuration information and return the value returned by that function. The calculation of the wrapping algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyWrapSymUpdate.

8.1.20.3 Csm_SymKeyWrapSymFinish

[SWS_Csm_00741] [

Service name:	Csm_SymKeyWrapSymFinish	
Syntax:	Std_ReturnType Csm_SymKeyWrapSymFinish(Csm_ConfigIdType cfgId)	
Service ID[hex]:	0x42	
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 interface shall be used to finish the symmetrical key wrapping 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 calculation of the wrapping algorithm is done by the underlying primitive.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyWrapSymFinish.

8.1.20.4 Csm_SymKeyWrapAsymStart

[SWS_Csm_00742] [

Service name:	Csm_SymKeyWrapAsymStart	

Syntax:	`Std_ReturnType Csm_SymKeyWrapAsymStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_SymKeyType* keyPtr,`
	`  const Csm_AsymPublicKeyType* wrappingKeyPtr`
	`)`	

Service ID[hex]:	0x43	
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 wrapping
	keyPtr	holds a pointer to the symmetric key to be wrapped
	wrappingKeyPtr	holds a pointer to the public key used for wrapping
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 key wrapping 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyWrapAsymStart.

8.1.20.5 Csm_SymKeyWrapAsymUpdate

[SWS_Csm_00743] [

Service name:	Csm_SymKeyWrapAsymUpdate

Syntax:	`Std_ReturnType Csm_SymKeyWrapAsymUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* dataPtr,`
	`  uint32* dataLengthPtr`
	`)`

Service ID[hex]:	0x44	
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):	dataLengthPtr	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 dataPtr. When the request has finished, the actual length of the computed value shall be stored.
Parameters (out):	dataPtr	holds a pointer to the memory location which will hold the first chunk of the result of the key wrapping. If the result does not fit into the given buffer, the caller shall call the service again, until *dataLengthPtr is equal to zero, indicating that the complete result has been retrieved.
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 retrieve the result of the key wrapping operation from the symmetrical key wrapping 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>Update of the primitive which is identified by the stored configuration information and return the value returned by that function. The calculation of the wrapping algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyWrapAsymUpdate.

8.1.20.6 Csm_SymKeyWrapAsymFinish

[SWS_Csm_00744] [

Service name:	Csm_SymKeyWrapAsymFinish	

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

Service ID[hex]:	0x45	
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 interface shall be used to finish the symmetrical key wrapping 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 calculation of the wrapping algorithm is done by the underlying	

	primitive.

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_SymKeyWrapAsymFinish.

8.1.21 Asymmetrical key update interface

Asymmetrical key update interface is used to update an asymmetrical key structure (e.g. public and private key pair) from certain data sources.

Note that these interfaces may be used for key transport purposes. In this case, any necessary auxiliary information (e.g., wrapping key, shared information, randomness) will have to be encoded unambiguously into the data provided in the dataPtr buffer.

8.1.21.1 Csm_AsymPrivateKeyUpdateStart

[SWS_Csm_00806] ⌈

Service name:	Csm_AsymPrivateKeyUpdateStart	
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyUpdateStart( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x5a	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the Asymmetric Private Key update.
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 asymmetric private key update 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyUpdateStart.

8.1.21.2 Csm_AsymPrivateKeyUpdateUpdate

[SWS_Csm_00807] ⌈

Service name:	Csm_AsymPrivateKeyUpdateUpdate
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyUpdateUpdate(`

	`Csm_ConfigIdType cfgId,`
	`const uint8* dataPtr,`
	`uint32 dataLength`
	)	

Service ID[hex]:	0x5b	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
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 a buffer containing one chunk of the key update data stream.
	dataLength	contains the length of the given chunk at dataPtr counted 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 interface shall be used to feed the Aymmetric Private Key Update 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 Asymmetric Private Key Update process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyUpdateUpdate.

8.1.21.3 Csm_AsymPrivateKeyUpdateFinish

[SWS_Csm_00808] [

Service name:	Csm_AsymPrivateKeyUpdateFinish	

Syntax:	`Std_ReturnType Csm_AsymPrivateKeyUpdateFinish(`
	`Csm_ConfigIdType cfgId,`
	`Csm_AsymPrivateKeyType* targetKeyPtr,`
	`uint8* proofPtr,`
	`uint32* proofLengthPtr`
	)	

Service ID[hex]:	0x5c	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	targetKeyPtr	holds a pointer to the target key buffer that may already contain a template for the asymmetric private key.
	proofLengthPtr	holds a pointer to the length associated with the buffer at proofPtr. The caller shall fill in the size of the provided buffer and CSM shall return the number of actually used bytes in it.
Parameters (out):	proofPtr	holds a pointer to a buffer that will hold the cryptographic proof that update has been successful.

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 Aymmetric Private Key Update 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 Asymmetric Private Key Update process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyUpdateFinish.

8.1.21.4 Csm_AsymPublicKeyUpdateStart

[SWS_Csm_00809] [

Service name:	Csm_AsymPublicKeyUpdateStart	
Syntax:	`Std_ReturnType Csm_AsymPublicKeyUpdateStart( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x57	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration which has to be used during the Asymmetric Public Key update.
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 Asymmetric key update 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPublicKeyUpdateStart.

8.1.21.5 Csm_AsymPublicKeyUpdateUpdate

[SWS_Csm_00810] [

Service name:	Csm_AsymPublicKeyUpdateUpdate	

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

Service ID[hex]:	0x58	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
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 a buffer containing one chunk of the key update data stream.
	dataLength	contains the length of the given chunk at dataPtr counted 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 interface shall be used to feed the Asymmetric Key Update 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 Asymmetric Public Key Update process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPublicKeyUpdateUpdate.

8.1.21.6 Csm_AsymPublicKeyUpdateFinish

[SWS_Csm_00811] [

Service name:	Csm_AsymPublicKeyUpdateFinish	

Syntax:	`Std_ReturnType Csm_AsymPublicKeyUpdateFinish(`
	`  Csm_ConfigIdType cfgId,`
	`  Csm_AsymPublicKeyType* targetKeyPtr,`
	`  uint8* proofPtr,`
	`  uint32* proofLengthPtr`
	`)`	

Service ID[hex]:	0x59	
Sync/Async:	Sync or Async, dependent on configuration (ECUC_Csm_00557)	
Reentrancy:	Non Reentrant	
Parameters (in):	cfgId	holds the identifier of the CSM module configuration that has to be used during the operation.
Parameters (inout):	targetKeyPtr	holds a pointer to the target key buffer that may already contain a template for the asymmetric key.
	proofLengthPtr	holds a pointer to the length associated with the buffer at proofPtr. The caller shall fill in the size of the provided buffer and CSM shall return the number of actually used bytes in it.

Parameters (out):	proofPtr	holds a pointer to a buffer that will hold the cryptographic proof that update has been successful.
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 Asymmetric Key Update 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 Asymmetric Public Key Update process is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPublicKeyUpdateFinish.

8.1.22 Asymmetrical key extract interfaces

Asymmetrical key extract interface is used to extract an asymmetrical key structure (e.g. public and private key pair) from certain data sources.

Note that these interfaces may be used for key transport purposes. In this case, any necessary auxiliary information (e.g., wrapping key, shared information, randomness) will have to be encoded unambiguously into the data provided in the dataPtr buffer.

8.1.22.1 Csm_AsymPublicKeyExtractStart

[SWS_Csm_00436] [

Service name:	Csm_AsymPublicKeyExtractStart	

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 interface shall be used to initialize the asymmetrical public key extraction 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPublicKeyExtractStart.

8.1.22.2 Csm_AsymPublicKeyExtractUpdate

[SWS_Csm_00443] [

Service name:	Csm_AsymPublicKeyExtractUpdate	
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 interface shall be used to feed the asymmetrical public key extraction service with 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 calculation of the extraction algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPublicKeyExtractUpdate.

8.1.22.3 Csm_AsymPublicKeyExtractFinish

[SWS_Csm_00450] [

Service name:	Csm_AsymPublicKeyExtractFinish
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 interface shall be used to finish the asymmetrical public key extraction 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 calculation of the extraction algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPublicKeyExtractFinish.

8.1.22.4 Csm_AsymPrivateKeyExtractStart

[SWS_Csm_00680] [

Service name:	Csm_AsymPrivateKeyExtractStart	
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyExtractStart( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x38	
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 interface shall be used to initialize the asymmetrical private key extraction 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".	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyExtractStart.

8.1.22.5 Csm_AsymPrivateKeyExtractUpdate

[SWS_Csm_00682] ⌈

Service name:	Csm_AsymPrivateKeyExtractUpdate	

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

Service ID[hex]:	0x39	
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 interface shall be used to feed the asymmetrical private key extraction service with 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 calculation of the extraction algorithm is done by the underlying primitive.	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyExtractUpdate.

8.1.22.6 Csm_AsymPrivateKeyExtractFinish

[SWS_Csm_00684] ⌈

Service name:	Csm_AsymPrivateKeyExtractFinish	

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

Service ID[hex]:	0x3a	
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 private 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 interface shall be used to finish the asymmetrical private key extraction 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 calculation of the extraction algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyExtractFinish.

8.1.23 Asymmetric key wrapping interface

Asymmetric key wrapping interface is used to export a (asymmetric) private key structure, e.g. to be used on a different device. To be able to use symmetric and asymmetric wrapping keys, two different interfaces are standardised.

8.1.23.1 Csm_AsymPrivateKeyWrapSymStart

[SWS_Csm_00745] [

Service name:	Csm_AsymPrivateKeyWrapSymStart	

Syntax:	`Std_ReturnType Csm_AsymPrivateKeyWrapSymStart(`
	`  Csm_ConfigIdType cfgId,`
	`  const Csm_AsymPrivateKeyType* keyPtr,`
	`  const Csm_SymKeyType* wrappingKeyPtr`
	`)`	

Service ID[hex]:	0x46	
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 wrapping
	keyPtr	holds a pointer to the private key to be wrapped
	wrappingKeyPtr	holds a pointer to the key used for wrapping
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 asymmetrical key wrapping 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyWrapSymStart.

8.1.23.2 Csm_AsymPrivateKeyWrapSymUpdate

[SWS_Csm_00746] [

Service name:	Csm_AsymPrivateKeyWrapSymUpdate	
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyWrapSymUpdate( Csm_ConfigIdType cfgId, uint8* dataPtr, uint32* dataLengthPtr )`	
Service ID[hex]:	0x47	
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):	dataLengthPtr	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 dataPtr. When the request has finished, the actual length of the computed value shall be stored.
Parameters (out):	dataPtr	holds a pointer to the memory location which will hold the first chunk of the result of the key wrapping. If the result does not fit into the given buffer, the caller shall call the service again, until *dataLengthPtr is equal to zero, indicating that the complete result has been retrieved.
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 retrieve the result of the key wrapping operation from the asymmetrical key wrapping 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>Update of the primitive which is identified by the stored configuration information and return the value returned by that function. The calculation of the wrapping algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyWrapSymUpdate.

8.1.23.3 Csm_AsymPrivateKeyWrapSymFinish

[SWS_Csm_00747] ⌈

Service name:	Csm_AsymPrivateKeyWrapSymFinish	
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyWrapSymFinish( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x48	
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 interface shall be used to finish the asymmetrical key wrapping 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 calculation of the wrapping algorithm is done by the underlying primitive.	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyWrapSymFinish.

8.1.23.4 Csm_AsymPrivateKeyWrapAsymStart

[SWS_Csm_00748] ⌈

Service name:	Csm_AsymPrivateKeyWrapAsymStart	
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyWrapAsymStart( Csm_ConfigIdType cfgId, const Csm_AsymPrivateKeyType* keyPtr, const Csm_AsymPublicKeyType* wrappingKeyPtr )`	
Service ID[hex]:	0x49	
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 wrapping
	keyPtr	holds a pointer to the private key to be wrapped
	wrappingKeyPtr	holds a pointer to the public key used for wrapping
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 asymmetrical key wrapping 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".

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyWrapAsymStart.

8.1.23.5 Csm_AsymPrivateKeyWrapAsymUpdate

[SWS_Csm_00749] [

Service name:	Csm_AsymPrivateKeyWrapAsymUpdate	

Syntax:	`Std_ReturnType Csm_AsymPrivateKeyWrapAsymUpdate(`
	`  Csm_ConfigIdType cfgId,`
	`  uint8* dataPtr,`
	`  uint32* dataLengthPtr`
	`)`	

Service ID[hex]:	0x4a	
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):	dataLengthPtr	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 dataPtr. When the request has finished, the actual length of the computed value shall be stored.
Parameters (out):	dataPtr	holds a pointer to the memory location which will hold the first chunk of the result of the key wrapping. If the result does not fit into the given buffer, the caller shall call the service again, until *dataLengthPtr is equal to zero, indicating that the complete result has been retrieved.
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 retrieve the result of the key wrapping operation from the asymmetrical key wrapping 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>Update of the primitive which is identified by the stored configuration information and return the value returned by that function. The calculation of the wrapping algorithm is done by the underlying primitive.	

] ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyWrapAsymUpdate.

8.1.23.6 Csm_AsymPrivateKeyWrapAsymFinish

[SWS_Csm_00750] ⌈

Service name:	Csm_AsymPrivateKeyWrapAsymFinish	
Syntax:	`Std_ReturnType Csm_AsymPrivateKeyWrapAsymFinish( Csm_ConfigIdType cfgId )`	
Service ID[hex]:	0x4b	
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 interface shall be used to finish the asymmetrical key wrapping 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 calculation of the wrapping algorithm is done by the underlying primitive.	

⌋ ()

Regarding error detection, the requirements **SWS_Csm_00488** and **SWS_Csm_00489** are applicable to the function Csm_AsymPrivateKeyWrapAsymFinish.

8.1.24 Dependencies to cryptographic library API functions

8.1.25 Types for the Cryptographic Primitives

8.1.25.1 Cry_<Primitive>ConfigType

[SWS_Csm_00544] ⌈

Name:	`Cry_<Primitive>ConfigType`		
Type:	`Structure`		
Element:	`void`	`implementation specific`	--
Description:	Data structure which shall encompass all information needed to specify the information needed for the <Primitive> cryptographic primitive.		

⌋ ()

8.1.26 API functions of the cryptographic primitives

[SWS_Csm_00461] ⌈ For every API function of a cryptographic service, the corresponding cryptographic primitive shall contain a corresponding function ⌋ (SRS_Csm_00006)

[SWS_Csm_00505] ⌈ The implementation of the basic cryptographic routines shall be synchronous or asynchronous, depending on the configuration of the CSM. ⌋ (SRS_Csm_00005)

8.1.26.1 Cry_<Primitive>Start

[SWS_Csm_00701] ⌈

Service name:	Cry_<Primitive>Start	

Syntax:	`Std_ReturnType Cry_<Primitive>Start(`
	`  <type> <xxx>`
	`)`	

Service ID[hex]:	--	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	<xxx>	The arguments <xxx> shall be identical to the arguments of the corresponding function Csm_<Service>Start(), with the exception of the argument cfgId. This argument is of type â€œCsm_ConfigIdTypeâ€œ in Csm_<Service>Start(). In Cry_<Primitive>Start the argument cfgId shall be replaced by an argument cfgPtr of type â€œconst void *â€œ.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	The return values shall be identical to those of the corresponding function Csm_<Service>Start().
Description:	Synchronous: This function shall initialize the computation of the cryptographic primitive, so that the primitive is able to process input data. Asynchronous: This function shall store the information given in the arguments, so that Cry_<Primitive>MainFunction() can process the initialisation.	

⌋ ()

[SWS_Csm_00732] ⌈
The API "Cry_<Primitive>Start" has a parameter "cfgPtr" of type "const void *".
⌋ ()

When calling this API, the parameter "cfgPtr" shall point to a constant variable of type "Cry_<Primitive>ConfigType", but shall be cast to "const void *".
Reason for this is to have a common definition of the parameter list of this API for all primitives of one service, because in the structure Csm_<Service>ConfigType one element is a function pointer to this API.

8.1.26.2 Cry_<Primitive>Update

[SWS_Csm_00702] ⌈

Service name:	Cry_<Primitive>Update	

Syntax:	`Std_ReturnType Cry_<Primitive>Update(`
	`  <type> <xxx>,`
	`  <type> <yyy>,`
	`  <type> <zzz>`
	`)`	

Service ID[hex]:	--	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	<xxx>	The arguments <xxx> shall be identical to the arguments of the corresponding function Csm_<Service>Update().
Parameters (inout):	<yyy>	The arguments <yyy> shall be identical to the arguments of the corresponding function Csm_<Service>Update().
Parameters (out):	<zzz>	The arguments <zzz> shall be identical to the arguments of the corresponding function Csm_<Service>Update().
Return value:	Std_ReturnType	The return values shall be identical to those of the corresponding function Csm_<Service>Update().
Description:	Synchronous: This function shall process a chunk of the given input data with the algorithm of the cryptographic primitive. Asynchronous: This function shall store the information given in the arguments, so that Cry_<Primitive>MainFunction() can process the input data. If the primitive is not ready to process the current request (e.g. because the processing of another Cry_<Primitive>Update has not yet finished), the function shall return with "CSM_E_BUSY".	

⌋ ()

8.1.26.3 Cry_<Primitive>Finish

[SWS_Csm_00703] ⌈

Service name:	Cry_<Primitive>Finish	

Syntax:	`Std_ReturnType Cry_<Primitive>Finish(`
	`  <type> <xxx>,`
	`  <type> <yyy>,`
	`  <type> <zzz>`
	`)`	

Service ID[hex]:	--	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	<xxx>	The arguments <xxx> shall be identical to the arguments of the corresponding function Csm_<Service>Finish().
Parameters (inout):	<yyy>	The arguments <yyy> shall be identical to the arguments of the corresponding function Csm_<Service>Finish().
Parameters (out):	<zzz>	The arguments <zzz> shall be identical to the arguments of the corresponding function Csm_<Service>Finish().
Return value:	Std_ReturnType	The return values shall be identical to those of the corresponding function Csm_<Service>Finish().
Description:	Synchronous: This function shall finish the computation of the cryptographic primitive and store the result into the memory location given. Asynchronous:	

	This function shall store the information given in the arguments, so that Cry_<Primitive>MainFunction() can finish the computation and store the result in the memory location given. If the primitive is not ready to process the current request (e.g. because the processing of a Cry_<Primitive>Update has not yet finished), the function has to return with "CSM_E_BUSY".

⌋ ()

8.1.26.4 Cry_<Primitive>

[SWS_Csm_00704] ⌈

Service name:	Cry_<Primitive>	

Syntax:	`Std_ReturnType Cry_<Primitive>(`
	`  <type> <xxx>`
	`)`	

Service ID[hex]:	--	
Sync/Async:	Sync or Async, dependent on configuration (CSM0557_Conf)	
Reentrancy:	Non Reentrant	
Parameters (in):	<xxx>	The arguments <xxx> shall be identical to the arguments of the corresponding function Csm_<Service>(), with the exception of the argument cfgId. This argument is of type â€œCsm_ConfigIdTypeâ€œ in Csm_<Service>(). In Cry_<Primitive> the argument cfgId shall be replaced by an argument cfgPtr of type â€œconst void *â€œ.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	The return values shall be identical to those of the corresponding function Csm_<Service>().
Description:	Synchronous: This function shall process the cryptographic primitive with the given input data and store the result in the memory location given. Asynchronous: This function shall prepare the computation of the cryptographic primitive, so that the primitive main function is able to process the input data and return the result.	

⌋ ()

[SWS_Csm_00733] ⌈ The API "Cry_<Primitive>" has a parameter "cfgPtr" of type "const void *".⌋ ()

When calling this API, the parameter "cfgPtr" shall point to a constant variable of type "Cry_<Primitive>ConfigType", but shall be cast to "const void *".
Reason for this is to have a common definition of the parameter list of this API for all primitives of one service, because in the structure Csm_<Service>ConfigType one element is a function pointer to this API.

8.1.26.5 Cry_<Primitive>MainFunction

[SWS_Csm_00773] ⌈

Service name:	Cry_<Primitive>MainFunction

Syntax:	`void Cry_<Primitive>MainFunction(`
	`  void`
	`)`

Service ID[hex]:	--

Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	The calculation of the cryptographic functions shall be done in the Cry_<Primitive>MainFunction(). When the main function has completely processed the cryptographic functions demanded by Cry_<Primitive>Start() or Cry_<Primitive>Update(), the corresponding callback Csm_<Service>CallbackNotification() must be called with the correct return value. When the main function has completely processed the cryptographic functions demanded by Cry_<Primitive>Finish(), the callback Csm_<Service>CallbackNotification() must be called with the correct return value and then the callback Csm_<Service>ServiceFinishNotification() must be called.

⌋ ()

8.1.27 Configuration of the cryptographic primitives

For each cryptographic primitive, a cryptographic library module has to provide a configuration structure. This configuration structure shall be of type `Cry_<Primitive>ConfigType`. For each configuration of a primitive, the cryptographic library module has to provide a constant variable of that type.
To link a primitive configuration to a specific service configuration, the corresponding parameter `Csm<Service>InitConfiguration` of the service configuration has to be set to the C-language symbol of the primitive configuration.

Variants of CRY modules with different optimization objectives may exist. These Variants should be handled by separate modules. Those optimizations may include execution speed, platform specific optimizations, RAM size and/or code segment size etc. The most suitable variant for a given deployment should be used.

8.1.28 Call-back notifications

8.1.29 CRY callback notifications

[SWS_Csm_00454] ⌈ When the cryptographic library has to change the main state machine of the CSM, this shall be done by using the following functions:
⌋ ()

8.1.29.1 Csm_<Service>CallbackNotification

[SWS_Csm_00455] ⌈

Service name:	Csm_<Service>CallbackNotification
Syntax:	`void Csm_<Service>CallbackNotification(`

	Std_ReturnType Result)
Service ID[hex]:	--
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Result: Contains the result of a cryptographic operation. 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 CSM_E_ENTROPY_EXHAUSTION: request failed, entropy of random number generator is exhausted.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	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)

8.1.29.2 Csm_<Service>ServiceFinishNotification

[SWS_Csm_00457] ⌈

Service name:	Csm_<Service>ServiceFinishNotification
Syntax:	void Csm_<Service>ServiceFinishNotification(void)
Service ID[hex]:	--
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This function shall set the state of the service <Service> to "idle".

⌋ (SRS_BSW_00359, SRS_BSW_00360)

8.1.30 User callback notifications

[SWS_Csm_00535] ⌈ User callback notifications are configured in the structure Csm_<Service>ConfigType (see SWS_Csm_0074). ⌋ ()

8.1.31 Scheduled functions

8.1.32 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 services.

] (SRS_BSW_00373, SRS_BSW_00432)

[SWS_Csm_00480] [This function shall perform the processing of the CSM module jobs.
] ()

[SWS_Csm_00469] [If a cryptographic service is active, the Csm_MainFunction() shall call the corresponding Cry_<Primitive>MainFunction() to calculate the cryptographic primitive.
] ()

[SWS_Csm_00483] [If no further job processing is possible, the Csm_MainFunction shall return immediately.
] ()

Interruption due to timeout supervision is necessary occasionally in case of involvement of cryptographic hardware only, what is under responsibility of specific drivers (out of scope of the CSM). This main function may be used for execution or triggering of any suiteable activity for hardwrare timeout supervision.

It may happen that an application becomes terminated before this application has been able to finish a sequence (Start, Update and Finish) of a dedicated CSM service. In such a case, the affected CSM service would remain in inits ‘active’ state and would not be usable any longer, until a RESET has been executed.
To avoid such, the CSM must perform some sanitary actions:

[SWS_Csm_00826] [The main function shall reset an ‘active’ CSM service to its ‘idle’ state, if two consecutive requests to start this already active service (by invocation of Csm_xxxStart) have been observed.
] ()

[SWS_Csm_00827] [If the main function resets a CSM service to its ‘idle’ state, the corresponding cryptographic primitive must be reset to its ‘idle’ state as well.
] ()

8.1.33 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.
] ()

8.2 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.

8.2.1 Client-Server-Interfaces

8.2.1.1 CsmHash

[SWS_Csm_00775] [

Name	CsmHash	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

HashFinish		
Comments	--	
Variation	--	
Parameters	resultBuffer	
	Comment	--
	Type	HashResultBuffer
	Variation	--

	Direction	OUT
	resultLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
	TruncationIsAllowed	
	Comment	--
	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		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
HashUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	HashDataBuffer
	Variation	--
	Direction	IN

	dataLength	
	Comment	--
	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)

8.2.1.2 CsmMacGenerate

[SWS_Csm_00776] ⌈

Name	CsmMacGenerate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

MacGenerateFinish		
Comments	--	
Variation	--	
Parameters	resultBuffer	
	Comment	--
	Type	MacGenerateResultBuffer
	Variation	--
	Direction	OUT
	resultLength	
	Comment	--
	Type	uint32

	Variation	--
	Direction	INOUT
	TruncationIsAllowed	
	Comment	--
	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		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
MacGenerateUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	MacGenerateDataBuffer
	Variation	--

	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.3 CsmMacVerify

[SWS_Csm_00777] [

Name	CsmMacVerify	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

MacVerifyFinish		
Comments	--	
Variation	--	
Parameters	MacBuffer	
	Comment	--
	Type	MacVerifyCompareBuffer
	Variation	--
	Direction	IN
	MacLength	
	Comment	--
	Type	uint32

	Variation	--
	Direction	IN
	resultBuffer	
	Comment	--
	Type	Csm_VerifyResultType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
MacVerifyStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
MacVerifyUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	MacVerifyDataBuffer
	Variation	--
	Direction	IN

	dataLength	
	Comment	--
	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)

8.2.1.4 CsmRandomSeed

[SWS_Csm_00778] ⌈

Name	CsmRandomSeed	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

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

RandomSeedStart		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

	CSM_E_BUSY	failed, service is still busy
RandomSeedUpdate		
Comments	--	
Variation	--	
Parameters	seedBuffer	
	Comment	--
	Type	RandomSeedDataBuffer
	Variation	--
	Direction	IN
	seedLength	
	Comment	--
	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)

8.2.1.5 CsmRandomGenerate

[SWS_Csm_00779] ⌈

Name	CsmRandomGenerate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	4	CSM_E_ENTROPY_EXHAUSTION

Operations

RandomGenerate

Comments	--	
Variation	--	
Parameters	resultBuffer	
	Comment	--
	Type	RandomGenerateResultBuffer
	Variation	--
	Direction	OUT
	resultLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
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_Csm_00066)

8.2.1.6 CsmSymBlockEncrypt

[SWS_Csm_00780] ⌈

Name	CsmSymBlockEncrypt	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymBlockEncryptFinish	
Comments	--

Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymBlockEncryptStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymBlockEncryptUpdate		
Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	SymBlockEncryptDataBuffer
	Variation	--
	Direction	IN
	plainTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	cipherTextBuffer	

	Comment	--
	Type	SymBlockEncryptResultBuffer
	Variation	--
	Direction	OUT
	cipherTextLength	
	Comment	--
	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)

8.2.1.7 CsmSymBlockDecrypt

[SWS_Csm_00781] [

Name	CsmSymBlockDecrypt	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

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

SymBlockDecryptStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymBlockDecryptUpdate		
Comments	--	
Variation	--	
Parameters	cipherTextBuffer	
	Comment	--
	Type	SymBlockDecryptDataBuffer
	Variation	--
	Direction	IN
	cipherTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	plainTextBuffer	
	Comment	--
	Type	SymBlockDecryptResultBuffer
	Variation	--
	Direction	OUT

	plainTextLength	
	Comment	--
	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)

8.2.1.8 CsmSymEncrypt

[SWS_Csm_00782] [

Name	CsmSymEncrypt	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymEncryptFinish		
Comments	--	
Variation	--	
Parameters	cipherTextBuffer	
	Comment	--
	Type	SymEncryptResultBuffer
	Variation	--
	Direction	OUT
	cipherTextLength	
	Comment	--

	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		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
	InitVectorBuffer	
	Comment	--
	Type	SymEncryptInitVectorBuffer
	Variation	--
	Direction	IN
	InitVectorLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymEncryptUpdate		

Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	SymEncryptDataBuffer
	Variation	--
	Direction	IN
	plainTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	cipherTextBuffer	
	Comment	--
	Type	SymEncryptResultBuffer
	Variation	--
	Direction	OUT
	cipherTextLength	
	Comment	--
	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)

8.2.1.9 CsmSymDecrypt

[SWS_Csm_00783] [

Name	CsmSymDecrypt
Comment	--

IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymDecryptFinish		
Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	SymDecryptResultBuffer
	Variation	--
	Direction	OUT
	plainTextLength	
	Comment	--
	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		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType

	Variation	--
	Direction	IN
	InitVectorBuffer	
	Comment	--
	Type	SymDecryptInitVectorBuffer
	Variation	--
	Direction	IN
	InitVectorLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymDecryptUpdate		
Comments	--	
Variation	--	
Parameters	cipherTextBuffer	
	Comment	--
	Type	SymDecryptDataBuffer
	Variation	--
	Direction	IN
	cipherTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	plainTextBuffer	
	Comment	--

	Type	SymDecryptResultBuffer
	Variation	--
	Direction	OUT
	plainTextLength	
	Comment	--
	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)

8.2.1.10 CsmAsymEncrypt

[SWS_Csm_00784]⌈

Name	CsmAsymEncrypt	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

AsymEncryptFinish		
Comments	--	
Variation	--	
Parameters	cipherTextBuffer	
	Comment	--
	Type	AsymEncryptResultBuffer

	Variation	--
	Direction	OUT
	cipherTextLength	
	Comment	--
	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		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_AsymPublicKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
AsymEncryptUpdate		
Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	AsymEncryptDataBuffer
	Variation	--

	Direction	IN
	plainTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	cipherTextBuffer	
	Comment	--
	Type	AsymEncryptResultBuffer
	Variation	--
	Direction	OUT
	cipherTextLength	
	Comment	--
	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)

8.2.1.11 CsmAsymDecrypt

[SWS_Csm_00785]⌈

Name	CsmAsymDecrypt	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

	3	CSM_E_SMALL_BUFFER

Operations

AsymDecryptFinish		
Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	AsymDecryptResultBuffer
	Variation	--
	Direction	OUT
	plainTextLength	
	Comment	--
	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		
Comments	--	
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		
Comments	--	
Variation	--	
Parameters	cipherTextBuffer	
	Comment	--
	Type	AsymDecryptDataBuffer
	Variation	--
	Direction	IN
	cipherTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	plainTextBuffer	
	Comment	--
	Type	AsymDecryptResultBuffer
	Variation	--
	Direction	OUT
	plainTextLength	
	Comment	--
	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)

8.2.1.12 CsmSignatureGenerate

[SWS_Csm_00786] ⌈

Name	CsmSignatureGenerate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SignatureGenerateFinish		
Comments	--	
Variation	--	
Parameters	resultBuffer	
	Comment	--
	Type	SignatureGenerateResultBuffer
	Variation	--
	Direction	OUT
	resultLength	
	Comment	--
	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	
Comments	--
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
SignatureGenerateUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	SignatureGenerateDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.13 CsmSignatureVerify

[SWS_Csm_00787] [

Name	CsmSignatureVerify
Comment	--
IsService	true
Variation	--

Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

SignatureVerifyFinish		
Comments	--	
Variation	--	
Parameters	signatureBuffer	
	Comment	--
	Type	SignatureVerifyCompareSignatureBuffer
	Variation	--
	Direction	IN
	signatureLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	resultBuffer	
	Comment	--
	Type	Csm_VerifyResultType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

SignatureVerifyStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--

	Type	Csm_AsymPublicKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SignatureVerifyUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	SignatureVerifyDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.14 CsmChecksum

[SWS_Csm_00788] ⌈

Name	CsmChecksum	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

ChecksumFinish		
Comments	--	
Variation	--	
Parameters	resultBuffer	
	Comment	--
	Type	ChecksumResultBuffer
	Variation	--
	Direction	OUT
	resultLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
	TruncationIsAllowed	
	Comment	--
	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
ChecksumStart		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful

	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
ChecksumUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	ChecksumDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.15 CsmKeyDerive

[SWS_Csm_00789] [

Name	CsmKeyDerive	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

KeyDeriveFinish

Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

KeyDeriveStart		
Comments	--	
Variation	--	
Parameters	keyLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
	iterations	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

KeyDeriveUpdate	
Comments	--
Variation	--

Parameters	passwordBuffer	
	Comment	--
	Type	KeyDerivePasswordBuffer
	Variation	--
	Direction	IN
	passwordLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	saltBuffer	
	Comment	--
	Type	KeyDeriveSaltBuffer
	Variation	--
	Direction	IN
	saltLength	
	Comment	--
	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)

8.2.1.16 CsmKeyDeriveSymKey

[SWS_Csm_00790] ⌈

Name	CsmKeyDeriveSymKey	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK

	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

KeyDeriveSymKey		
Comments	--	
Variation	--	
Parameters	baseKey	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
	customisationValBuffer	
	Comment	--
	Type	KeyDeriveSymKeyCustBuffer
	Variation	--
	Direction	IN
	customisationValLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	derivedKey	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

] (SRS_Csm_00066)

8.2.1.17 CsmKeyExchangeCalcPubVal

[SWS_Csm_00791] ⌈

Name	CsmKeyExchangeCalcPubVal	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

KeyExchangeCalcPubVal		
Comments	--	
Variation	--	
Parameters	baseBuffer	
	Comment	--
	Type	Csm_KeyExchangeBaseType
	Variation	--
	Direction	IN
	privateValueBuffer	
	Comment	--
	Type	Csm_KeyExchangePrivateType
	Variation	--
	Direction	IN
	publicValueBuffer	
	Comment	--
	Type	KeyExchangeOwnPublicValueBuffer
	Variation	--
	Direction	OUT
	publicValueLength	
	Comment	--

	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)

8.2.1.18 CsmKeyExchangeCalcSecret

[SWS_Csm_00792] [

Name	CsmKeyExchangeCalcSecret	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

KeyExchangeCalcSecretFinish		
Comments	--	
Variation	--	
Parameters	sharedSecretBuffer	
	Comment	--
	Type	KeyExchangeSharedSecretBuffer
	Variation	--
	Direction	OUT
	sharedSecretLength	
	Comment	--
	Type	uint32
	Variation	--

	Direction	INOUT
	TruncationIsAllowed	
	Comment	--
	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

KeyExchangeCalcSecretStart	
Comments	--
Variation	--

Parameters	baseBuffer	
	Comment	--
	Type	Csm_KeyExchangeBaseType
	Variation	--
	Direction	IN
	privateValueBuffer	
	Comment	--
	Type	Csm_KeyExchangePrivateType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

KeyExchangeCalcSecretUpdate	
Comments	--
Variation	--

Parameters	partnerPublicValueBuffer	
	Comment	--
	Type	KeyExchangePartnerPublicValueBuffer
	Variation	--
	Direction	IN
	partnerPublicValueLength	
	Comment	--
	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)

8.2.1.19 CsmKeyExchangeCalcSymKey

[SWS_Csm_00793] [

Name	CsmKeyExchangeCalcSymKey	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

KeyExchangeCalcSymKeyFinish		
Comments	--	
Variation	--	
Parameters	sharedSecretLength	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--

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

KeyExchangeCalcSymKeyStart		
Comments	--	
Variation	--	
Parameters	baseBuffer	
	Comment	--
	Type	Csm_KeyExchangeBaseType
	Variation	--
	Direction	IN
	privateValueBuffer	
	Comment	--
	Type	Csm_KeyExchangePrivateType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

KeyExchangeCalcSymKeyUpdate		
Comments	--	
Variation	--	
Parameters	partnerPublicValueBuffer	
	Comment	--
	Type	KeyExchangePartnerPublicValueBuffer
	Variation	--
	Direction	IN
	partnerPublicValueLength	

	Comment	--
	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)

8.2.1.20 CsmSymKeyExtract

[SWS_Csm_00794] [

Name	CsmSymKeyExtract	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

SymKeyExtractFinish		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymKeyExtractStart		

Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymKeyExtractUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	SymKeyExtractDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.21 CsmSymKeyWrapSym

[SWS_Csm_00795] ⌈

Name	CsmSymKeyWrapSym	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

	2	CSM_E_BUSY

Operations

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

SymKeyWrapSymStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
	wrappingKey	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

SymKeyWrapSymUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--

	Type	SymKeyWrapSymDataBuffer
	Variation	--
	Direction	OUT
	dataLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

] (SRS_Csm_00066)

8.2.1.22 CsmSymKeyWrapAsym

[SWS_Csm_00796] [

Name	CsmSymKeyWrapAsym	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

SymKeyWrapAsymFinish		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymKeyWrapAsymStart		
Comments	--	

Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN
	wrappingKey	
	Comment	--
	Type	Csm_AsymPublicKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymKeyWrapAsymUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	SymKeyWrapAsymDataBuffer
	Variation	--
	Direction	OUT
	dataLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

⌋ (SRS_Csm_00066)

8.2.1.23 CsmAsymPublicKeyExtract

[SWS_Csm_00797] ⌈

Name	CsmAsymPublicKeyExtract	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

AsymPublicKeyExtractFinish		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_AsymPublicKeyType
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

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

AsymPublicKeyExtractUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	AsymPublicKeyExtractDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.24 CsmAsymPrivateKeyExtract

[SWS_Csm_00798]⌈

Name	CsmAsymPrivateKeyExtract	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

AsymPrivateKeyExtractFinish	
Comments	--
Variation	--
Parameters	key

	Comment	--
	Type	Csm_AsymPrivateKeyType
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

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

AsymPrivateKeyExtractUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	AsymPrivateKeyExtractDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.25 CsmAsymPrivateKeyWrapSym

[SWS_Csm_00799] ⌈

Name	CsmAsymPrivateKeyWrapSym	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

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

AsymPrivateKeyWrapSymStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_AsymPrivateKeyType
	Variation	--
	Direction	IN
	wrappingKey	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	IN

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
AsymPrivateKeyWrapSymUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	AsymPrivateKeyWrapSymDataBuffer
	Variation	--
	Direction	OUT
	dataLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

] (SRS_Csm_00066)

8.2.1.26 CsmAsymPrivateKeyWrapAsym

[SWS_Csm_00800] [

Name	CsmAsymPrivateKeyWrapAsym	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

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

AsymPrivateKeyWrapAsymStart		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_AsymPrivateKeyType
	Variation	--
	Direction	IN
	wrappingKey	
	Comment	--
	Type	Csm_AsymPublicKeyType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

AsymPrivateKeyWrapAsymUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	AsymPrivateKeyWrapAsymDataBuffer
	Variation	--

	Direction	OUT
	dataLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	INOUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy

] (SRS_Csm_00066)

8.2.1.27 CsmSymKeyGenerate

[SWS_Csm_00805] [

Name	CsmSymKeyGenerate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY

Operations

SymKeyGenerate		
Comments	--	
Variation	--	
Parameters	keyBuffer	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	INOUT
	keyLength	
	Comment	--
	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)

8.2.1.28 CsmAsymPublicKeyUpdate

[SWS_Csm_00812] [

Name	CsmAsymPublicKeyUpdate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

AsymPublicKeyUpdateFinish		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_AsymPublicKeyType
	Variation	--
	Direction	INOUT
	proofBuffer	
	Comment	--
	Type	ProofDataBuffer
	Variation	--
	Direction	OUT
	proofLength	

	Comment	--
	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
AsymPublicKeyUpdateStart		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
AsymPublicKeyUpdateUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	AsymPublicKeyUpdateDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.29 CsmAsymPrivateKeyUpdate

[SWS_Csm_00813] [

Name	CsmAsymPrivateKeyUpdate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

AsymPrivateKeyUpdateFinish		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_AsymPrivateKeyType
	Variation	--
	Direction	INOUT
	proofBuffer	
	Comment	--
	Type	ProofDataBuffer
	Variation	--
	Direction	OUT
	proofLength	
	Comment	--
	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
AsymPrivateKeyUpdateStart		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
AsymPrivateKeyUpdateUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	AsymPrivateKeyUpdateDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.30 CsmSymKeyUpdate

[SWS_Csm_00817] [

Name	CsmSymKeyUpdate	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

SymKeyUpdateFinish		
Comments	--	
Variation	--	
Parameters	key	
	Comment	--
	Type	Csm_SymKeyType
	Variation	--
	Direction	INOUT
	proofBuffer	
	Comment	--
	Type	ProofDataBuffer
	Variation	--
	Direction	OUT
	proofLength	
	Comment	--
	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

SymKeyUpdateStart		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	CSM_E_BUSY	failed, service is still busy
SymKeyUpdateUpdate		
Comments	--	
Variation	--	
Parameters	dataBuffer	
	Comment	--
	Type	SymKeyUpdateDataBuffer
	Variation	--
	Direction	IN
	dataLength	
	Comment	--
	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)

8.2.1.31 CsmCompress

[SWS_Csm_00824] ⌈

Name	CsmCompress
Comment	--
IsService	true

Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

CompressFinish		
Comments	--	
Variation	--	
Parameters	compressedTextBuffer	
	Comment	--
	Type	CompressResultBuffer
	Variation	--
	Direction	OUT
	compressedTextLength	
	Comment	--
	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

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

CompressUpdate		
Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	CompressDataBuffer
	Variation	--
	Direction	IN
	plainTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	compressedTextBuffer	
	Comment	--
	Type	CompressResultBuffer
	Variation	--
	Direction	OUT
	compressedTextLength	
	Comment	--
	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)

8.2.1.32 CsmDecompress

[SWS_Csm_00825] [

Name	CsmDecompress

Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	CSM_E_BUSY
	3	CSM_E_SMALL_BUFFER

Operations

DecompressFinish		
Comments	--	
Variation	--	
Parameters	decompressedTextBuffer	
	Comment	--
	Type	DecompressResultBuffer
	Variation	--
	Direction	OUT
	decompressedTextLength	
	Comment	--
	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
DecompressStart		
Comments	--	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

	CSM_E_BUSY	failed, service is still busy
DecompressUpdate		
Comments	--	
Variation	--	
Parameters	plainTextBuffer	
	Comment	--
	Type	DecompressDataBuffer
	Variation	--
	Direction	IN
	plainTextLength	
	Comment	--
	Type	uint32
	Variation	--
	Direction	IN
	decompressedTextBuffer	
	Comment	--
	Type	DecompressResultBuffer
	Variation	--
	Direction	OUT
	decompressedTextLength	
	Comment	--
	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)

To access the configured callback functions, the CSM provides one Port per configuration. This Port is assigned to a Client/Server-Interface, which contains

[SWS_Csm_00801] ⌈

Name	CsmCallback	
Comment	--	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

JobFinished		
Comments	--	
Variation	--	
Parameters	retVal	
	Comment	--
	Type	Std_ReturnType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

⌋ (SRS_Csm_00066)

8.2.2 Implementation Data Types

[SWS_Csm_00831]⌈

Name	Csm_AlignType
Kind	Type
Derived from	<maxAlignScalarType>
Description	A scalar type which has maximum alignment restrictions on the given platform. This value is configured by "CsmMaxAlignScalarType". <maxAlignScalarType> can be e.g. uint8, uint16 or uint32. All context buffers shall be aligned according to the maximum alignment of all scalar types on

	the given platform.
Variation	--

] (SRS_Csm_00066)

[SWS_Csm_00832][

Name	Csm_CallbackType
Kind	Type
Derived from	Std_FunctionPointer
Description	Function pointer for the callback function which will be invoked after service completion. Signature: Std_ReturnType (*Csm_CallbackType)(Std_ReturnType)
Variation	--

] (SRS_Csm_00066)

[SWS_Csm_00833][

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)

[SWS_Csm_00834][

Name	Csm_VerifyResultType		
Kind	Type		
Derived from	uint8		
Description	Enumeration of the result type of verification operations.		
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)

[SWS_Csm_00835][

Name	Csm_AsymPublicKeyType		
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_PUB_KEY_MAX_SIZE
Description	Structure for the public asymmetrical key. CSM_ASYM_PUB_KEY_MAX_SIZE shall be chosen such that "CSM_ASYM_PUB_KEY_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmAsymEncryptMaxKeySize, CsmSignatureVerifyMaxKeySize, CsmAsymPublicKeyExtractMaxKeySize, CsmSymKeyWrapAsymMaxPubKeySize, CsmAsymPrivateKeyWrapAsymPubKeySize and CsmAsymPublicKeyUpdateMaxKeySize.		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00836][

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_Csm_00066)

[SWS_Csm_00837][

Name	Csm_SymKeyType		
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_SYM_KEY_MAX_SIZE
Description	Structure for the symmetrical key. CSM_SYM_KEY_MAX_SIZE shall be chosen such that "CSM_SYM_KEY_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmSymBlockEncryptMaxKeySize, CsmSymBlockDecryptMaxKeySize, CsmSymEncryptMaxKeySize, CsmSymDecryptMaxKeySize, CsmKeyDeriveMaxKeySize, CsmSymKeyExtractMaxKeySize, CsmMacGenerateMaxKeySize, CsmMacVerifyMaxKeySize, CsmSymKeyWrapSymMaxSymKeySize, CsmSymKeyWrapAsymMaxSymKeySize, CsmAsymPrivateKeyWrapSymMaxSymKeySize, CsmKeyExchangeCalcSymKeyMaxSymKeySize, CsmKeyDeriveSymKeyMaxSymKeySize, CsmSymKeyUpdateMaxKeySize and CsmSymKeyGenerateMaxKeySize.		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00838][

Name	Csm_KeyExchangeBaseType		
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_KEY_EX_BASE_MAX_SIZE
Description	Structure with base type information of the key exchange protocol. CSM_KEY_EX_BASE_MAX_SIZE shall be chosen such that "CSM_KEY_EX_BASE_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmKeyExchangeCalcPubValMaxBaseTypeSize, CsmKeyExchangeCalcSecretMaxBaseTypeSize and CsmKeyExchangeCalcSymKeyMaxBaseTypeSize.		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00839][

Name	Csm_KeyExchangePrivateType
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_KEY_EX_PRIV_MAX_SIZE
Description	Structure with the private Information of the key exchange protocol only known to the current user. CSM_KEY_EX_PRIV_MAX_SIZE shall be chosen such that "CSM_KEY_EX_PRIV_MAX_SIZE * sizeof(Csm_AlignType)" is greater or equal to the maximum of the configured values CsmKeyExchangeCalcPubValMaxPrivateTypeSize, CsmKeyExchangeCalcSecretMaxPrivateTypeSize CsmKeyExchangeCalcSymKeyMaxPrivateTypeSize.		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00840][

Name	AsymDecryptDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00841][

Name	AsymDecryptResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00842][

Name	AsymEncryptDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00843][

Name	AsymEncryptResultBuffer		
Kind	Array	Element type	uint8

Description	--
Variation	--

] (SRS_Csm_00066)

[SWS_Csm_00844][

Name	AsymPrivateKeyExtractDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00845][

Name	AsymPrivateKeyUpdateDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00846][

Name	AsymPrivateKeyWrapAsymDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00847][

Name	AsymPrivateKeyWrapSymDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00848][

Name	AsymPublicKeyExtractDataBuffer		
Kind	Array	Element type	uint8
Description	--		

Variation	--

⌋ (SRS_Csm_00066)

[SWS_Csm_00849]⌈

Name	AsymPublicKeyUpdateDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋(SRS_Csm_00066)

[SWS_Csm_00850]⌈

Name	ChecksumDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00851]⌈

Name	ChecksumResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00852]⌈

Name	CompressDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00853]⌈

Name	CompressResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00854][

Name	DecompressDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00855][

Name	DecompressResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00856][

Name	HashDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00857][

Name	HashResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00858][

Name	KeyDerivePasswordBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00859]⌈

Name	KeyDeriveSaltBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00860]⌈

Name	KeyDeriveSymKeyCustBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00861]⌈

Name	KeyExchangeOwnPublicValueBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00862]⌈

Name	KeyExchangePartnerPublicValueBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00863]⌈

Name	KeyExchangeSharedSecretBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00864]⌈

Name	MacGenerateDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00865][

Name	MacGenerateResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00866][

Name	MacVerifyCompareBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00867][

Name	MacVerifyDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00868][

Name	ProofDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00869][

Name	RandomGenerateResultBuffer

Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00870][

Name	RandomSeedDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00871][

Name	SignatureGenerateDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00872][

Name	SignatureGenerateResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00873][

Name	SignatureVerifyCompareSignatureBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00874][

Name	SignatureVerifyDataBuffer		
Kind	Array	Element type	uint8

Description	--
Variation	--

] (SRS_Csm_00066)

[SWS_Csm_00875][

Name	SymBlockDecryptDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00876][

Name	SymBlockDecryptResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00877][

Name	SymBlockEncryptDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00878][

Name	SymDecryptDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00879][

Name	SymDecryptInitVectorBuffer		
Kind	Array	Element type	uint8
Description	--		

Variation	--

] (SRS_Csm_00066)

[SWS_Csm_00880][

Name	SymDecryptResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00881][

Name	SymEncryptDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00882][

Name	SymEncryptInitVectorBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00883][

Name	SymEncryptResultBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

] (SRS_Csm_00066)

[SWS_Csm_00884][

Name	SymKeyExtractDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00885]⌈

Name	SymKeyUpdateDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00886]⌈

Name	SymKeyWrapAsymDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

[SWS_Csm_00887]⌈

Name	SymKeyWrapSymDataBuffer		
Kind	Array	Element type	uint8
Description	--		
Variation	--		

⌋ (SRS_Csm_00066)

8.2.3 Ports

[SWS_Csm_00888]⌈

Name	{Name}_AsymDecrypt		
Kind	ProvidedPort	Interface	CsmAsymDecrypt
Description	--		
Variation	Name = {ecuc(Csm/CsmAsymDecrypt/CsmAsymDecryptConfig.SHORT-NAME)}		

⌋(SRS_Csm_00066)

[SWS_Csm_00889]⌈

Name	{Name}_AsymEncrypt		
Kind	ProvidedPort	Interface	CsmAsymEncrypt
Description	--		

Variation	Name = {ecuc(Csm/CsmAsymEncrypt/CsmAsymEncryptConfig.SHORT-NAME)}

] (SRS_Csm_00066)

[SWS_Csm_00890][

Name	{Name}_AsymPrivateKeyExtract		
Kind	ProvidedPort	Interface	CsmAsymPrivateKeyExtract
Description	--		
Variation	Name = {ecuc(Csm/CsmAsymPrivateKeyExtract/CsmAsymPrivateKeyExtractConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00891][

Name	{Name}_AsymPrivateKeyUpdate		
Kind	ProvidedPort	Interface	CsmAsymPrivateKeyUpdate
Description	--		
Variation	Name = {ecuc(Csm/CsmAsymPrivateKeyUpdate/CsmAsymPrivateKeyUpdateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00892][

Name	{Name}_AsymPrivateKeyWrapAsym		
Kind	ProvidedPort	Interface	CsmAsymPrivateKeyWrapAsym
Description	--		
Variation	Name = {ecuc(Csm/CsmAsymPrivateKeyWrapAsym/ CsmAsymPrivateKeyWrapAsymConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00893][

Name	{Name}_AsymPrivateKeyWrapSym		
Kind	ProvidedPort	Interface	CsmAsymPrivateKeyWrapSym
Description	--		
Variation	Name = {ecuc(Csm/CsmAsymPrivateKeyWrapSym/ CsmAsymPrivateKeyWrapSymConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00894][

Name	{Name}_AsymPublicKeyExtract		
Kind	ProvidedPort	Interface	CsmAsymPublicKeyExtract

Description	--
Variation	Name = {ecuc(Csm/CsmAsymPublicKeyExtract/CsmAsymPublicKeyExtractConfig.SHORT-NAME)}

] (SRS_Csm_00066)

[SWS_Csm_00895][

Name	{Name}_AsymPublicKeyUpdate		
Kind	ProvidedPort	Interface	CsmAsymPublicKeyUpdate
Description	--		
Variation	Name = {ecuc(Csm/CsmAsymPublicKeyUpdate/CsmAsymPublicKeyUpdateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00896][

Name	{Name}_Callback		
Kind	RequiredPort	Interface	CsmCallback
Description	--		
Variation	Name = {ecuc(Csm/*/*Config.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00897][

Name	{Name}_Checksum		
Kind	ProvidedPort	Interface	CsmChecksum
Description	--		
Variation	Name = {ecuc(Csm/CsmChecksum/CsmChecksumConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00898][

Name	{Name}_Compress		
Kind	ProvidedPort	Interface	CsmCompress
Description	--		
Variation	Name = {ecuc(Csm/CsmCompression/CsmCompressionConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00899][

Name	{Name}_Decompress		
Kind	ProvidedPort	Interface	CsmDecompress

Description	--
Variation	Name = {ecuc(Csm/CsmDecompression/CsmDecompressionConfig.SHORT-NAME)}

] (SRS_Csm_00066)

[SWS_Csm_00900][

Name	{Name}_Hash		
Kind	ProvidedPort	Interface	CsmHash
Description	--		
Variation	Name = {ecuc(Csm/CsmHash/CsmHashConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00901][

Name	{Name}_KeyDerive		
Kind	ProvidedPort	Interface	CsmKeyDerive
Description	--		
Variation	Name = {ecuc(Csm/CsmKeyDerive/CsmKeyDeriveConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00902][

Name	{Name}_KeyDeriveSymKey		
Kind	ProvidedPort	Interface	CsmKeyDeriveSymKey
Description	--		
Variation	Name = {ecuc(Csm/CsmKeyDeriveSymKey/CsmKeyDeriveSymKeyConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00903][

Name	{Name}_KeyExchangeCalcPubVal		
Kind	ProvidedPort	Interface	CsmKeyExchangeCalcPubVal
Description	--		
Variation	Name = {ecuc(Csm/CsmKeyExchangeCalcPubVal/ CsmKeyExchangeCalcPubValConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00904][

Name	{Name}_KeyExchangeCalcSecret		
Kind	ProvidedPort	Interface	CsmKeyExchangeCalcSecret

Description	--
Variation	Name = {ecuc(Csm/CsmKeyExchangeCalcSecret/CsmKeyExchangeCalcSecretConfig.SHORT-NAME)}

] (SRS_Csm_00066)

[SWS_Csm_00905][

Name	{Name}_KeyExchangeCalcSymKey		
Kind	ProvidedPort	Interface	CsmKeyExchangeCalcSymKey
Description	--		
Variation	Name = {ecuc(Csm/CsmKeyExchangeCalcSymKey/CsmKeyExchangeCalcSymKeyConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00906][

Name	{Name}_MacGenerate		
Kind	ProvidedPort	Interface	CsmMacGenerate
Description	--		
Variation	Name = {ecuc(Csm/CsmMacGenerate/CsmMacGenerateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00907][

Name	{Name}_MacVerify		
Kind	ProvidedPort	Interface	CsmMacVerify
Description	--		
Variation	Name = {ecuc(Csm/CsmMacVerify/CsmMacVerifyConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00908][

Name	{Name}_RandomGenerate		
Kind	ProvidedPort	Interface	CsmRandomGenerate
Description	--		
Variation	Name = {ecuc(Csm/CsmRandomGenerate/CsmRandomGenerateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00909][

Name	{Name}_RandomSeed

Kind	ProvidedPort	Interface	CsmRandomSeed
Description	--		
Variation	Name = {ecuc(Csm/CsmRandomSeed/CsmRandomSeedConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00910][

Name	{Name}_SignatureGenerate		
Kind	ProvidedPort	Interface	CsmSignatureGenerate
Description	--		
Variation	Name = {ecuc(Csm/CsmSignatureGenerate/CsmSignatureGenerateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00911][

Name	{Name}_SignatureVerify		
Kind	ProvidedPort	Interface	CsmSignatureVerify
Description	--		
Variation	Name = {ecuc(Csm/CsmSignatureVerify/CsmSignatureVerifyConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00913][

Name	{Name}_SymBlockDecrypt		
Kind	ProvidedPort	Interface	CsmSymBlockDecrypt
Description	--		
Variation	Name = {ecuc(Csm/CsmSymBlockDecrypt/CsmSymBlockDecryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00914][

Name	{Name}_SymBlockEncrypt		
Kind	ProvidedPort	Interface	CsmSymBlockEncrypt
Description	--		
Variation	Name = {ecuc(Csm/CsmSymBlockEncrypt/CsmSymBlockEncryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00915][

Name	{Name}_SymDecrypt		
Kind	ProvidedPort	Interface	CsmSymDecrypt
Description	--		
Variation	Name = {ecuc(Csm/CsmSymDecrypt/CsmSymDecryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00916][

Name	{Name}_SymEncrypt		
Kind	ProvidedPort	Interface	CsmSymEncrypt
Description	--		
Variation	Name = {ecuc(Csm/CsmSymEncrypt/CsmSymEncryptConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00917][

Name	{Name}_SymKeyExtract		
Kind	ProvidedPort	Interface	CsmSymKeyExtract
Description	--		
Variation	Name = {ecuc(Csm/CsmSymKeyExtract/CsmSymKeyExtractConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00918][

Name	{Name}_SymKeyGenerate		
Kind	ProvidedPort	Interface	CsmSymKeyGenerate
Description	--		
Variation	Name = {ecuc(Csm/CsmSymKeyGenerate/CsmSymKeyGenerateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00919][

Name	{Name}_SymKeyUpdate		
Kind	ProvidedPort	Interface	CsmSymKeyUpdate
Description	--		
Variation	Name = {ecuc(Csm/CsmSymKeyUpdate/CsmSymKeyUpdateConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00920][

Name	{Name}_SymKeyWrapAsym		
Kind	ProvidedPort	Interface	CsmSymKeyWrapAsym
Description	--		
Variation	Name = {ecuc(Csm/CsmSymKeyWrapAsym/CsmSymKeyWrapAsymConfig.SHORT-NAME)}		

] (SRS_Csm_00066)

[SWS_Csm_00921][

Name	{Name}_SymKeyWrapSym		
Kind	ProvidedPort	Interface	CsmSymKeyWrapSym
Description	--		
Variation	Name = {ecuc(Csm/CsmSymKeyWrapSym/CsmSymKeyWrapSymConfig.SHORT-NAME)}		

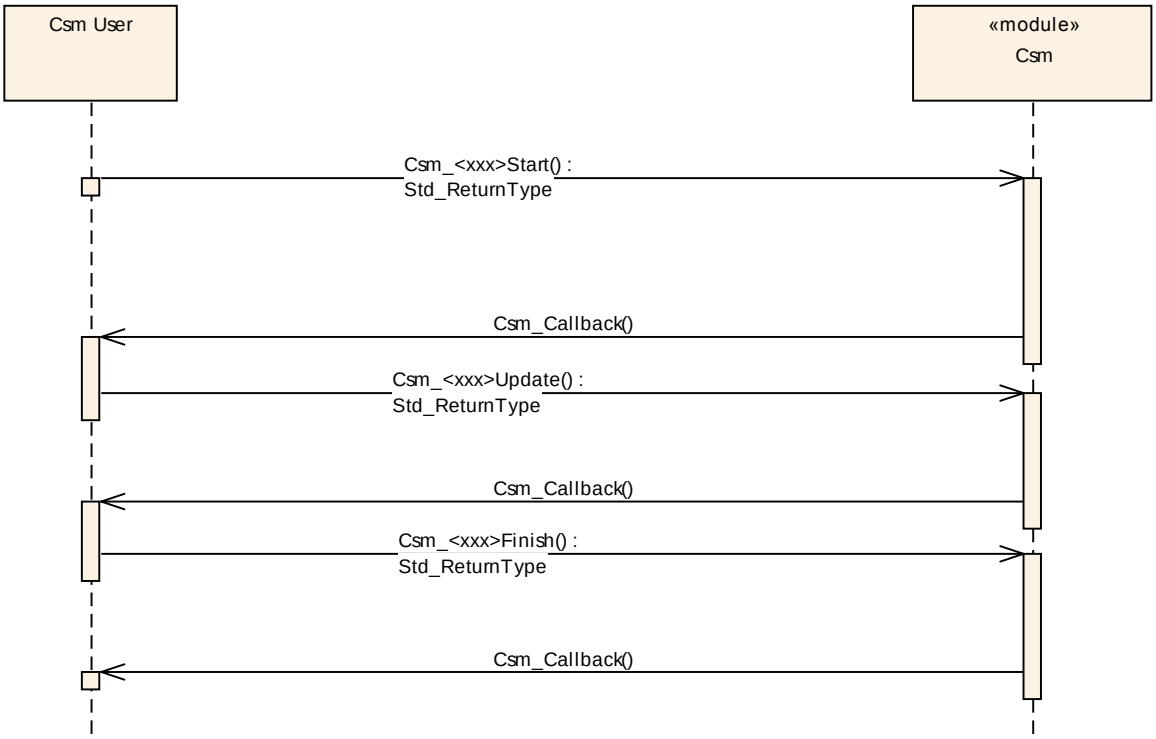
] (SRS_Csm_00066)

9 Sequence diagrams

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

9.1 Asynchronous calls

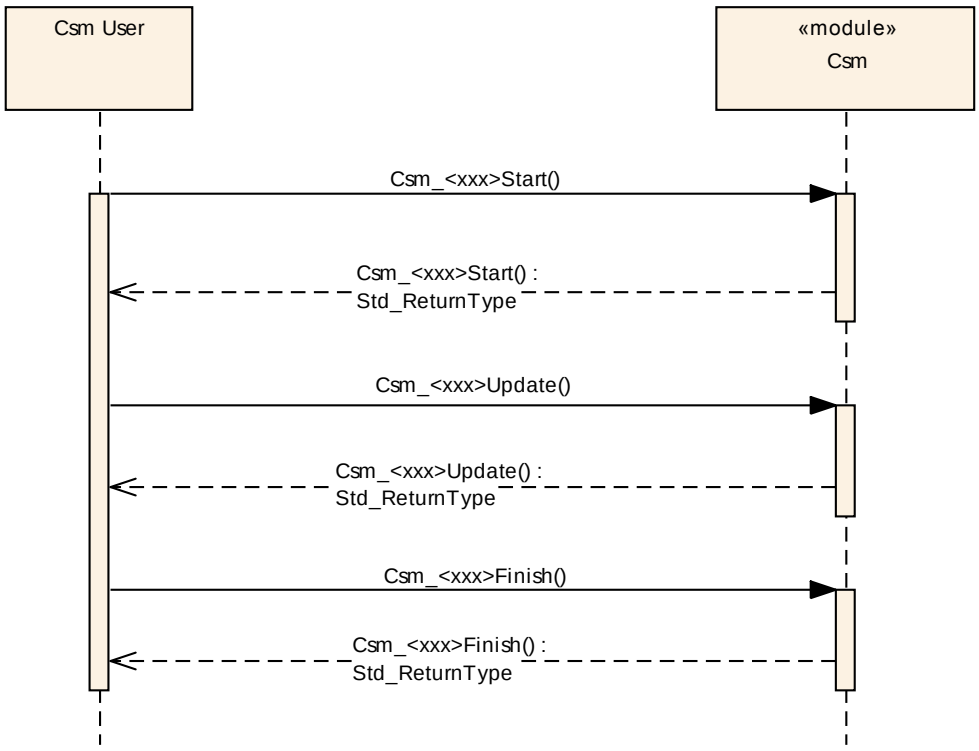
The following diagram (Sequence diagram for asynchronous call) shows a sample sequence of function calls for a request which is performed asynchronously. The result of the asynchronous function can be accessed after an asynchronous notification (invocation of the configured callback function).



Sequence diagram for asynchronous call with callback

9.2 Synchronous calls

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

10.2.1 Variants

For details refer to the chapter 10.1.2 “Variants” in *SWS_BSWGeneral.*

10.2.2 Csm

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymDecrypt	0..1	Container for incorporation of AsymDecrypt primitives.
CsmAsymEncrypt	0..1	Container for incorporation of AsymEncrypt primitives.
CsmAsymPrivateKeyExtract	0..1	Container for incorporation of AsymPrivateKeyExtract primitives.
CsmAsymPrivateKeyUpdate	0..1	Container for incorporation of Asymmetric Private Key Update primitives.
CsmAsymPrivateKeyWrapAsym	0..1	Container for incorporation of AsymPrivateKeyWrapSym primitives.
CsmAsymPrivateKeyWrapSym	0..1	Container for incorporation of AsymPrivateKeyWrapSym primitives.
CsmAsymPublicKeyExtract	0..1	Container for incorporation of AsymPublicKeyExtract primitives.
CsmAsymPublicKeyUpdate	0..1	Container for incorporation of Asymmetric Public Key

		Update primitives.
CsmChecksum	0..1	Container for incorporation of Checksum primitives.
CsmCompression	0..1	Container for incorporation of Compression primitives.
CsmDecompression	0..1	Container for incorporation of Decompression primitives.
CsmGeneral	1	Container for common configuration options.
CsmHash	0..1	Container for incorporation of Hash primitives.
CsmKeyDerive	0..1	Container for incorporation of KeyDerive primitives.
CsmKeyDeriveSymKey	0..1	Container for incorporation of CsmKeyDeriveSymKey primitives.
CsmKeyExchangeCalcPubVal	0..1	Container for incorporation of KeyExchangeCalcPubVal primitives.
CsmKeyExchangeCalcSecret	0..1	Container for incorporation of KeyExchangeCalcSecret primitives.
CsmKeyExchangeCalcSymKey	0..1	Container for incorporation of KeyExchangeCalcSymKey primitives.
CsmMacGenerate	0..1	Container for incorporation of MacGenerate primitives.
CsmMacVerify	0..1	Container for incorporation of MacVerify primitives.
CsmRandomGenerate	0..1	Container for incorporation of RandomGenerate primitives.
CsmRandomSeed	0..1	Container for incorporation of RandomSeed primitives.
CsmSignatureGenerate	0..1	Container for incorporation of SignatureGenerate primitives.
CsmSignatureVerify	0..1	Container for incorporation of SignatureVerify primitives.
CsmSymBlockDecrypt	0..1	Container for incorporation of SymBlockDecrypt primitives.
CsmSymBlockEncrypt	0..1	Container for incorporation of SymBlockEncrypt primitives.
CsmSymDecrypt	0..1	Container for incorporation of SymDecrypt primitives.
CsmSymEncrypt	0..1	Container for incorporation of SymEncrypt primitives.
CsmSymKeyExtract	0..1	Container for incorporation of SymKeyExtract primitives.
CsmSymKeyGenerate	0..1	Container for incorporation of Symmetric Key Generate primitives.
CsmSymKeyUpdate	0..1	Container for incorporation of Symmetric Key Update primitives.
CsmSymKeyWrapAsym	0..1	Container for incorporation of SymKeyWrapSym primitives.
CsmSymKeyWrapSym	0..1	Container for incorporation of SymKeyWrapSym primitives.

10.2.3 CsmGeneral

SWS Item	**ECUC_Csm_00554 :**
Container Name	CsmGeneral
Description	Container for common configuration options.
Configuration Parameters	

SWS Item	**ECUC_Csm_00555 :**		
Name	CsmDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_Csm_00773 :**		
Name	CsmMainFunctionPeriod		
Description	Specifies the period of main function Csm_MainFunction in seconds.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	-INF .. INF		
Default value	--		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00729 :**		
Name	CsmMaxAlignScalarType		
Description	The scalar type which has the maximum alignment restrictions on the given platform. This type can be e.g. uint8, uint16 or uint32.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00557 :**		
Name	CsmUseSyncJobProcessing		
Description	Pre-processor switch to enable and disable synchronous job processing. True: synchronous job processing enabled False: synchronous job processing disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00708 :**
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.4 CsmHash

SWS Item	**ECUC_Csm_00559 :**
Container Name	CsmHash
Description	Container for incorporation of Hash primitives.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmHashConfig	0..32	Configurations for the Hash service

10.2.5 CsmHashConfig

SWS Item	**ECUC_Csm_00560 :**
Container Name	CsmHashConfig
Description	Configurations for the Hash service. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00561 :**		
Name	CsmCallbackHash		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00563 :**
Name	CsmHashInitConfiguration

Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00562 :**		
Name	CsmHashPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.6 CsmMacGenerate

SWS Item	**ECUC_Csm_00635 :**
Container Name	CsmMacGenerate
Description	Container for incorporation of MacGenerate primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00709 :**		
Name	CsmMacGenerateMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a MAC generation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmMacGenerateConfig	0..32	Configurations for the MacGenerate service.

10.2.7 CsmMacGenerateConfig

SWS Item	**ECUC_Csm_00564 :**
Container Name	CsmMacGenerateConfig
Description	Configurations for the MacGenerate service. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00565 :**		
Name	CsmCallbackMacGenerate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00567 :**		
Name	CsmMacGenerateInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00566 :**
Name	CsmMacGeneratePrimitiveName
Description	Name of the cryptographic primitive to use.
Multiplicity	1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--

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

No Included Containers

10.2.8 CsmMacVerify

SWS Item	**ECUC_Csm_00636 :**
Container Name	CsmMacVerify
Description	Container for incorporation of MacVerify primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00710 :**		
Name	CsmMacVerifyMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a MAC verification.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmMacVerifyConfig	0..32	Configurations for the MacVerify service

10.2.9 CsmMacVerifyConfig

SWS Item	**ECUC_Csm_00568 :**
Container Name	CsmMacVerifyConfig
Description	Container for configuration of service MacVerify. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00569 :**
Name	CsmCallbackMacVerify
Description	Callback function to be called if service has finished.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	--
maxLength	--
minLength	--

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

SWS Item	**ECUC_Csm_00571 :**		
Name	CsmMacVerifyInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00570 :**		
Name	CsmMacVerifyPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.10 CsmRandomSeed

SWS Item	**ECUC_Csm_00641 :**
Container Name	CsmRandomSeed
Description	Container for incorporation of RandomSeed primitives.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmRandomSeedConfig	0..32	Configurations for the RandomSeed service

10.2.11 CsmRandomSeedConfig

SWS Item	**ECUC_Csm_00642 :**		
Container Name	CsmRandomSeedConfig		
Description	Container for configuration of service RandomSeed. The container name serves as a symbolic name for the identifier of a service configuration.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00643 :**		
Name	CsmCallbackRandomSeed		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00645 :**		
Name	CsmRandomSeedInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00644 :**		
Name	CsmRandomSeedPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.12 CsmRandomGenerate

SWS Item	**ECUC_Csm_00620 :**
Container Name	CsmRandomGenerate
Description	Container for incorporation of RandomGenerate primitives.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmRandomGenerateConfig	0..32	Configurations for the RandomGenerate service

10.2.13 CsmRandomGenerateConfig

SWS Item	**ECUC_Csm_00637 :**
Container Name	CsmRandomGenerateConfig
Description	Container for configuration of service RandomGenerate. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00638 :**		
Name	CsmCallbackRandomGenerate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00640 :**		
Name	CsmRandomGenerateInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_Csm_00639 :**		
Name	CsmRandomGeneratePrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.14 CsmSymBlockEncrypt

SWS Item	**ECUC_Csm_00621 :**
Container Name	CsmSymBlockEncrypt
Description	Container for incorporation of SymBlockEncrypt primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00711 :**		
Name	CsmSymBlockEncryptMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical block encryption.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymBlockEncryptConfig	0..32	Configurations for the SymBlockEncrypt service

10.2.15 CsmSymBlockEncryptConfig

SWS Item	**ECUC_Csm_00572 :**
Container Name	CsmSymBlockEncryptConfig
Description	Container for configuration of service SymBlockEncrypt. The container

	name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00573 :**		
Name	CsmCallbackSymBlockEncrypt		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00575 :**		
Name	CsmSymBlockEncryptInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00574 :**		
Name	CsmSymBlockEncryptPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.16 CsmSymBlockDecrypt

SWS Item	**ECUC_Csm_00622 :**		
Container Name	CsmSymBlockDecrypt		
Description	Container for incorporation of SymBlockDecrypt primitives.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00712 :**		
Name	CsmSymBlockDecryptMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical block decryption.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymBlockDecryptConfig	0..32	Configurations for the SymBlockDecrypt service

10.2.17 CsmSymBlockDecryptConfig

SWS Item	**ECUC_Csm_00576 :**
Container Name	CsmSymBlockDecryptConfig
Description	Container for configuration of service SymBlockDecrypt. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00577 :**		
Name	CsmCallbackSymBlockDecrypt		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00579 :**
Name	CsmSymBlockDecryptInitConfiguration
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.

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

SWS Item	**ECUC_Csm_00578 :**		
Name	CsmSymBlockDecryptPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.18 CsmSymEncrypt

SWS Item	**ECUC_Csm_00623 :**
Container Name	CsmSymEncrypt
Description	Container for incorporation of SymEncrypt primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00713 :**		
Name	CsmSymEncryptMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical encryption.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***

CsmSymEncryptConfig	0..32	Configurations for the SymEncrypt service

10.2.19 CsmSymEncryptConfig

SWS Item	**ECUC_Csm_00580 :**
Container Name	CsmSymEncryptConfig
Description	Container for configuration of service SymEncrypt. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00581 :**		
Name	CsmCallbackSymEncrypt		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00583 :**		
Name	CsmSymEncryptInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00582 :**		
Name	CsmSymEncryptPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

No Included Containers

10.2.20 CsmSymDecrypt

SWS Item	**ECUC_Csm_00624 :**
Container Name	CsmSymDecrypt
Description	Container for incorporation of SymDecrypt primitives
Configuration Parameters	

SWS Item	**ECUC_Csm_00714 :**		
Name	CsmSymDecryptMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical decryption.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymDecryptConfig	0..32	Configurations for the SymDecrypt service.

10.2.21 CsmSymDecryptConfig

SWS Item	**ECUC_Csm_00584 :**
Container Name	CsmSymDecryptConfig
Description	Container for configuration of service SymDecrypt. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00585 :**
Name	CsmCallbackSymDecrypt
Description	Callback function to be called if service has finished.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	--
maxLength	--
minLength	--
regularExpression	--
Post-Build Variant Value	false

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

SWS Item	**ECUC_Csm_00587 :**		
Name	CsmSymDecryptInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00586 :**		
Name	CsmSymDecryptPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.22 CsmAsymEncrypt

SWS Item	**ECUC_Csm_00625 :**
Container Name	CsmAsymEncrypt
Description	Container for incorporation of AsymEncrypt primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00715 :**	
Name	CsmAsymEncryptMaxKeySize	
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical encryption.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymEncryptConfig	0..32	Configurations for the AsymEncrypt service

10.2.23 CsmAsymEncryptConfig

SWS Item	**ECUC_Csm_00588 :**
Container Name	CsmAsymEncryptConfig
Description	Container for configuration of service AsymEncrypt. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00591 :**		
Name	CsmAsymEncryptInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00590 :**		
Name	CsmAsymEncryptPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00589 :**
Name	CsmCallbackAsymEncrypt

Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.24 CsmAsymDecrypt

SWS Item	**ECUC_Csm_00626 :**
Container Name	CsmAsymDecrypt
Description	Container for incorporation of AsymDecrypt primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00716 :**		
Name	CsmAsymDecryptMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical decryption.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymDecryptConfig	0..32	Configurations for the AsymDecrypt service

10.2.25 CsmAsymDecryptConfig

SWS Item	**ECUC_Csm_00592 :**
Container Name	CsmAsymDecryptConfig
Description	Container for configuration of service AsymDecrypt. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00595 :**

Name	CsmAsymDecryptInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00594 :**		
Name	CsmAsymDecryptPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00593 :**		
Name	CsmCallbackAsymDecrypt		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.26 CsmSignatureGenerate

SWS Item	**ECUC_Csm_00627 :**
Container Name	CsmSignatureGenerate
Description	Container for incorporation of SignatureGenerate primitives

Configuration Parameters

SWS Item	ECUC_Csm_00717 :		
Name	CsmSignatureGenerateMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a signature generation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
CsmSignatureGenerateConfig	0..32	Configurations for the SignatureGenerate service

10.2.27 CsmSignatureGenerateConfig

SWS Item	ECUC_Csm_00596 :
Container Name	CsmSignatureGenerateConfig
Description	Container for configuration of service SignatureGenerate. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	ECUC_Csm_00597 :		
Name	CsmCallbackSignatureGenerate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_Csm_00599 :
Name	CsmSignatureGenerateInitConfiguration
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.
Multiplicity	1
Type	EcucStringParamDef
Default value	--
maxLength	--

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

SWS Item	**ECUC_Csm_00598 :**		
Name	CsmSignatureGeneratePrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.28 CsmSignatureVerify

SWS Item	**ECUC_Csm_00628 :**
Container Name	CsmSignatureVerify
Description	Container for incorporation of SignatureVerify primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00718 :**		
Name	CsmSignatureVerifyMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a signature verification.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSignatureVerifyConfig	0..32	Configurations for the SignatureVerify service

10.2.29 CsmSignatureVerifyConfig

SWS Item	**ECUC_Csm_00600 :**		
Container Name	CsmSignatureVerifyConfig		
Description	Container for configuration of service SignatureVerify. The container name serves as a symbolic name for the identifier of a service configuration.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00601 :**		
Name	CsmCallbackSignatureVerify		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00603 :**		
Name	CsmSignatureVerifyInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00602 :**		
Name	CsmSignatureVerifyPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.30 CsmCompression

SWS Item	**ECUC_Csm_00807 :**
Container Name	CsmCompression
Description	Container for incorporation of Compression primitives.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmCompressionConfig	0..32	Container for configuration of service Compression. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.31 CsmCompressionConfig

SWS Item	**ECUC_Csm_00808 :**
Container Name	CsmCompressionConfig
Description	Container for configuration of service Compression. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00809 :**		
Name	CsmCallbackCompression		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00810 :**		
Name	CsmCompressionInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_Csm_00811 :**		
Name	CsmCompressionPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.32 CsmDecompression

SWS Item	**ECUC_Csm_00812 :**
Container Name	CsmDecompression
Description	Container for incorporation of Decompression primitives.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmDecompressionConfig	0..32	Container for configuration of service Decompression. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.33 CsmDecompressionConfig

SWS Item	**ECUC_Csm_00813 :**
Container Name	CsmDecompressionConfig
Description	Container for configuration of service Decompression. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00814 :**
Name	CsmCallbackDecompression
Description	Callback function to be called if service has finished.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	--
maxLength	--

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

SWS Item	**ECUC_Csm_00815 :**		
Name	CsmDecompressionInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00816 :**		
Name	CsmDecompressionPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.34 CsmChecksum

SWS Item	**ECUC_Csm_00629 :**
Container Name	CsmChecksum
Description	Container for incorporation of Checksum primitives.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmChecksumConfig	0..32	Configurations for the Checksum service

10.2.35 CsmChecksumConfig

SWS Item	**ECUC_Csm_00604 :**		
Container Name	CsmChecksumConfig		
Description	Container for configuration of service Checksum. The container name serves as a symbolic name for the identifier of a service configuration.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00605 :**		
Name	CsmCallbackChecksum		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00607 :**		
Name	CsmChecksumInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00606 :**		
Name	CsmChecksumPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

No Included Containers

10.2.36 CsmKeyDerive

SWS Item	**ECUC_Csm_00630 :**
Container Name	CsmKeyDerive
Description	Container for incorporation of KeyDerive primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00719 :**		
Name	CsmKeyDeriveMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a key derivation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmKeyDeriveConfig	0..32	Configurations for the KeyDerive service

10.2.37 CsmKeyDeriveConfig

SWS Item	**ECUC_Csm_00608 :**
Container Name	CsmKeyDeriveConfig
Description	Container for configuration of service KeyDerive. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00609 :**		
Name	CsmCallbackKeyDerive		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_Csm_00611 :**		
Name	CsmKeyDeriveInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00610 :**		
Name	CsmKeyDerivePrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.38 CsmKeyExchangeCalcPubVal

SWS Item	**ECUC_Csm_00631 :**
Container Name	CsmKeyExchangeCalcPubVal
Description	Container for incorporation of KeyExchangeCalcPubVal primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00720 :**	
Name	CsmKeyExchangeCalcPubValMaxBaseTypeSize	
Description	The maximum length, in bytes, of all base types used in all CRY primitives which implement a public value calculation.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	--	

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

SWS Item	**ECUC_Csm_00721 :**		
Name	CsmKeyExchangeCalcPubValMaxPrivateTypeSize		
Description	The maximum length, in bytes, of all private information types used in all CRY primitives which implement a public value calculation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmKeyExchangeCalcPubValConfig	0..32	Configurations for the KeyExchangeCalcPubVal service

10.2.39 CsmKeyExchangeCalcPubValConfig

SWS Item	**ECUC_Csm_00612 :**
Container Name	CsmKeyExchangeCalcPubValConfig
Description	Container for configuration of service KeyExchangeCalcPubVal. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00613 :**		
Name	CsmCallbackKeyExchangeCalcPubVal		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00615 :**
Name	CsmKeyExchangeCalcPubValInitConfiguration
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.

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

SWS Item	**ECUC_Csm_00614 :**		
Name	CsmKeyExchangeCalcPubValPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.40 CsmKeyExchangeCalcSecret

SWS Item	**ECUC_Csm_00632 :**
Container Name	CsmKeyExchangeCalcSecret
Description	Container for incorporation of KeyExchangeCalcSecret primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00722 :**		
Name	CsmKeyExchangeCalcSecretMaxBaseTypeSize		
Description	The maximum length, in bytes, of all base types used in all CRY primitives which implement a shared secret calculation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00723 :**
Name	CsmKeyExchangeCalcSecretMaxPrivateTypeSize

Description	The maximum length, in bytes, of all private information types used in all CRY primitives which implement a shared secret calculation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmKeyExchangeCalcSecretConfig	0..32	Configurations for the KeyExchangeCalcSecret service.

10.2.41 CsmKeyExchangeCalcSecretConfig

SWS Item	**ECUC_Csm_00616 :**
Container Name	CsmKeyExchangeCalcSecretConfig
Description	Container for configuration of service KeyExchangeCalcSecret. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00617 :**		
Name	CsmCallbackKeyExchangeCalcSecret		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00545 :**		
Name	CsmKeyExchangeCalcSecretInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_Csm_00618 :**		
Name	CsmKeyExchangeCalcSecretPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.42 CsmSymKeyGenerate

SWS Item	**ECUC_Csm_00783 :**		
Container Name	CsmSymKeyGenerate		
Description	Container for incorporation of Symmetric Key Generate primitives.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00803 :**		
Name	CsmSymKeyGenerateMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical key generation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymKeyGenerateConfig	0..32	Container for configuration of service Symmetric Key Generate. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.43 CsmSymKeyGenerateConfig

SWS Item	**ECUC_Csm_00784 :**		
Container Name	CsmSymKeyGenerateConfig		
Description	Container for configuration of service Symmetric Key Generate. The container name serves as a symbolic name for the identifier of a service configuration.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00785 :**		
Name	CsmCallbackSymKeyGenerate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00786 :**		
Name	CsmSymKeyGenerateInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00787 :**		
Name	CsmSymKeyGeneratePrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.44 CsmSymKeyExtract

SWS Item	**ECUC_Csm_00633 :**		
Container Name	CsmSymKeyExtract		
Description	Container for incorporation of SymKeyExtract primitives.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00724 :**		
Name	CsmSymKeyExtractMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical key extraction.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymKeyExtractConfig	0..32	Configurations for the SymKeyExtract service

10.2.45 CsmSymKeyExtractConfig

SWS Item	**ECUC_Csm_00546 :**
Container Name	CsmSymKeyExtractConfig
Description	Container for configuration of service SymKeyExtract. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00547 :**		
Name	CsmCallbackSymKeyExtract		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00549 :**
Name	CsmSymKeyExtractInitConfiguration

Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00548 :**		
Name	CsmSymKeyExtractPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.46 CsmSymKeyUpdate

SWS Item	**ECUC_Csm_00788 :**
Container Name	CsmSymKeyUpdate
Description	Container for incorporation of Symmetric Key Update primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00804 :**		
Name	CsmSymKeyUpdateMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical key update.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***

CsmSymKeyUpdateConfig	0..32	Container for configuration of service Symmetric Key Update. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.47 CsmSymKeyUpdateConfig

SWS Item	**ECUC_Csm_00789 :**
Container Name	CsmSymKeyUpdateConfig
Description	Container for configuration of service Symmetric Key Update. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00790 :**		
Name	CsmCallbackSymKeyUpdate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00791 :**		
Name	CsmSymKeyUpdateInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00792 :**
Name	CsmSymKeyUpdatePrimitiveName
Description	Name of the cryptographic primitive to use.
Multiplicity	1
Type	EcucStringParamDef
Default value	--
maxLength	--
minLength	--
regularExpression	--

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

No Included Containers

10.2.48 CsmAsymPublicKeyUpdate

SWS Item	**ECUC_Csm_00793 :**
Container Name	CsmAsymPublicKeyUpdate
Description	Container for incorporation of Asymmetric Public Key Update primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00805 :**		
Name	CsmAsymPublicKeyUpdateMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical public key update.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymPublicKeyUpdateConfig	0..32	Container for configuration of service Asymmetric Public Key Update. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.49 CsmAsymPublicKeyUpdateConfig

SWS Item	**ECUC_Csm_00794 :**
Container Name	CsmAsymPublicKeyUpdateConfig
Description	Container for configuration of service Asymmetric Public Key Update. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00796 :**
Name	CsmAsymPublicKeyUpdateInitConfiguration
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.
Multiplicity	1
Type	EcucStringParamDef
Default value	--

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

SWS Item	**ECUC_Csm_00797 :**		
Name	CsmAsymPublicKeyUpdatePrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00795 :**		
Name	CsmCallbackAsymPublicKeyUpdate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.50 CsmAsymPrivateKeyUpdate

SWS Item	**ECUC_Csm_00798 :**
Container Name	CsmAsymPrivateKeyUpdate
Description	Container for incorporation of Asymmetric Private Key Update primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00806 :**
Name	CsmAsymPrivateKeyUpdateMaxKeySize
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical private key update.

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymPrivateKeyUpdateConfig	0..32	Container for configuration of service Asymmetric Private Key Update. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.51 CsmAsymPrivateKeyUpdateConfig

SWS Item	**ECUC_Csm_00799 :**
Container Name	CsmAsymPrivateKeyUpdateConfig
Description	Container for configuration of service Asymmetric Private Key Update. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00801 :**		
Name	CsmAsymPrivateKeyUpdateInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00802 :**		
Name	CsmAsymPrivateKeyUpdatePrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

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

SWS Item	**ECUC_Csm_00800 :**		
Name	CsmCallbackAsymPrivateKeyUpdate		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.52 CsmAsymPublicKeyExtract

SWS Item	**ECUC_Csm_00634 :**
Container Name	CsmAsymPublicKeyExtract
Description	Container for incorporation of AsymPublicKeyExtract primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00725 :**		
Name	CsmAsymPublicKeyExtractMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical public key extraction.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymPublicKeyExtractConfig	0..32	Configurations for the AsymPublicKeyExtract service.

10.2.53 CsmAsymPublicKeyExtractConfig

SWS Item	**ECUC_Csm_00550 :**

Container Name	CsmAsymPublicKeyExtractConfig
Description	Container for configuration of service AsymPublicKeyExtract. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00553 :**		
Name	CsmAsymPublicKeyExtractInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00552 :**		
Name	CsmAsymPublicKeyExtractPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00551 :**		
Name	CsmCallbackAsymPublicKeyExtract		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.54 CsmAsymPrivateKeyExtract

SWS Item	**ECUC_Csm_00686 :**		
Container Name	CsmAsymPrivateKeyExtract		
Description	Container for incorporation of AsymPrivateKeyExtract primitives.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00726 :**		
Name	CsmAsymPrivateKeyExtractMaxKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical private key extraction.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymPrivateKeyExtractConfig	0..32	Configurations for the AsymPrivateKeyExtract service.

10.2.55 CsmAsymPrivateKeyExtractConfig

SWS Item	**ECUC_Csm_00687 :**
Container Name	CsmAsymPrivateKeyExtractConfig
Description	Container for configuration of service AsymPrivateKeyExtract. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00690 :**		
Name	CsmAsymPrivateKeyExtractInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00689 :**
Name	CsmAsymPrivateKeyExtractPrimitiveName

Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00688 :**		
Name	CsmCallbackAsymPrivateKeyExtract		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.56 CsmKeyExchangeCalcSymKey

SWS Item	**ECUC_Csm_00732 :**
Container Name	CsmKeyExchangeCalcSymKey
Description	Container for incorporation of KeyExchangeCalcSymKey primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00738 :**		
Name	CsmKeyExchangeCalcSymKeyMaxBaseTypeSize		
Description	The maximum length, in bytes, of all base types used in all CRY primitives which implement a symmetrical key calculation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00737 :**

Name	CsmKeyExchangeCalcSymKeyMaxPrivateTypeSize		
Description	The maximum length, in bytes, of all private information types used in all CRY primitives which implement a symmetrical key calculation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00739 :**		
Name	CsmKeyExchangeCalcSymKeyMaxSymKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a symmetrical key calculation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmKeyExchangeCalcSymKeyConfig	0..32	Container for configuration of service KeyExchangeCalcSymKey. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.57 CsmKeyExchangeCalcSymKeyConfig

SWS Item	**ECUC_Csm_00736 :**
Container Name	CsmKeyExchangeCalcSymKeyConfig
Description	Container for configuration of service KeyExchangeCalcSymKey. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00733 :**
Name	CsmCallbackKeyExchangeCalcSymKey
Description	Callback function to be called if service has finished.
Multiplicity	1
Type	EcucFunctionNameDef
Default value	--
maxLength	--
minLength	--
regularExpression	--
Post-Build Variant Value	false

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

SWS Item	**ECUC_Csm_00735 :**		
Name	CsmKeyExchangeCalcSymKeyInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00734 :**		
Name	CsmKeyExchangeCalcSymKeyPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.58 CsmAsymPrivateKeyWrapSym

SWS Item	**ECUC_Csm_00752 :**
Container Name	CsmAsymPrivateKeyWrapSym
Description	Container for incorporation of AsymPrivateKeyWrapSym primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00758 :**	
Name	CsmAsymPrivateKeyWrapSymMaxPrivKeySize	
Description	The maximum length, in bytes, of all private information types used in all CRY primitives which implement an asymmetric private key wrapping.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	1 .. 4294967295	
Default value	--	
Post-Build Variant Value	false	

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

SWS Item	**ECUC_Csm_00757 :**		
Name	CsmAsymPrivateKeyWrapSymMaxSymKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetric private key wrapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymPrivateKeyWrapSymConfig	0..32	Container for configuration of service AsymPrivateKeyWrapSym. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.59 CsmAsymPrivateKeyWrapSymConfig

SWS Item	**ECUC_Csm_00753 :**
Container Name	CsmAsymPrivateKeyWrapSymConfig
Description	Container for configuration of service AsymPrivateKeyWrapSym. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00756 :**		
Name	CsmAsymPrivateKeyWrapSymInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00755 :**
Name	CsmAsymPrivateKeyWrapSymPrimitiveName
Description	Name of the cryptographic primitive to use.

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

SWS Item	**ECUC_Csm_00754 :**		
Name	CsmCallbackAsymPrivateKeyWrapSym		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.60 CsmAsymPrivateKeyWrapAsym

SWS Item	**ECUC_Csm_00759 :**
Container Name	CsmAsymPrivateKeyWrapAsym
Description	Container for incorporation of AsymPrivateKeyWrapSym primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00765 :**		
Name	CsmAsymPrivateKeyWrapAsymMaxPrivKeySize		
Description	The maximum length, in bytes, of all private key types used in all CRY primitives which implement an asymmetrical private key wrapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00764 :**
Name	CsmAsymPrivateKeyWrapAsymMaxPubKeySize
Description	The maximum length, in bytes, of all public key types used in all CRY primitives which implement an asymmetrical private key wrapping.

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmAsymPrivateKeyWrapAsymConfig	0..32	Container for configuration of service SymKeyWrapAsym. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.61 CsmAsymPrivateKeyWrapAsymConfig

SWS Item	**ECUC_Csm_00760 :**
Container Name	CsmAsymPrivateKeyWrapAsymConfig
Description	Container for configuration of service SymKeyWrapAsym. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00763 :**		
Name	CsmAsymPrivateKeyWrapAsymInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00762 :**		
Name	CsmAsymPrivateKeyWrapAsymPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

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

SWS Item	**ECUC_Csm_00761 :**		
Name	CsmCallbackAsymPrivateKeyWrapAsym		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.62 CsmSymKeyWrapSym

SWS Item	**ECUC_Csm_00740 :**		
Container Name	CsmSymKeyWrapSym		
Description	Container for incorporation of SymKeyWrapSym primitives.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00745 :**		
Name	CsmSymKeyWrapSymMaxSymKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an symmetrical key wrapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymKeyWrapSymConfig	0..32	Container for configuration of service SymKeyWrapSym. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.63 CsmSymKeyWrapSymConfig

SWS Item	**ECUC_Csm_00741 :**
Container Name	CsmSymKeyWrapSymConfig
Description	Container for configuration of service SymKeyWrapSym. The container

	name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00742 :**		
Name	CsmCallbackSymKeyWrapSym		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00744 :**		
Name	CsmSymKeyWrapSymInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00743 :**		
Name	CsmSymKeyWrapSymPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.64 CsmSymKeyWrapAsym

SWS Item	**ECUC_Csm_00746 :**		
Container Name	CsmSymKeyWrapAsym		
Description	Container for incorporation of SymKeyWrapSym primitives.		
Configuration Parameters			

SWS Item	**ECUC_Csm_00766 :**		
Name	CsmSymKeyWrapAsymMaxPubKeySize		
Description	The maximum length, in bytes, of all public key types used in all CRY primitives which implement an asymmetrical key wrapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00748 :**		
Name	CsmSymKeyWrapAsymMaxSymKeySize		
Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement an asymmetrical key wrapping.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmSymKeyWrapAsymConfig	0..32	Container for configuration of service SymKeyWrapAsym. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.65 CsmSymKeyWrapAsymConfig

SWS Item	**ECUC_Csm_00747 :**
Container Name	CsmSymKeyWrapAsymConfig
Description	Container for configuration of service SymKeyWrapAsym. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00749 :**
Name	CsmCallbackSymKeyWrapAsym
Description	Callback function to be called if service has finished.
Multiplicity	1

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

SWS Item	**ECUC_Csm_00751 :**		
Name	CsmSymKeyWrapAsymInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00750 :**		
Name	CsmSymKeyWrapAsymPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.66 CsmKeyDeriveSymKey

SWS Item	**ECUC_Csm_00767 :**
Container Name	CsmKeyDeriveSymKey
Description	Container for incorporation of CsmKeyDeriveSymKey primitives.
Configuration Parameters	

SWS Item	**ECUC_Csm_00769 :**
Name	CsmKeyDeriveSymKeyMaxSymKeySize

Description	The maximum, in bytes, of all key lengths used in all CRY primitives which implement a key derivation.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 4294967295		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CsmKeyDeriveSymKeyConfig	0..32	Container for configuration of service CsmKeyDeriveSymKey. The container name serves as a symbolic name for the identifier of a service configuration.

10.2.67 CsmKeyDeriveSymKeyConfig

SWS Item	**ECUC_Csm_00768 :**
Container Name	CsmKeyDeriveSymKeyConfig
Description	Container for configuration of service CsmKeyDeriveSymKey. The container name serves as a symbolic name for the identifier of a service configuration.
Configuration Parameters	

SWS Item	**ECUC_Csm_00770 :**		
Name	CsmCallbackKeyDeriveSymKey		
Description	Callback function to be called if service has finished.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_Csm_00772 :**		
Name	CsmKeyDeriveSymKeyInitConfiguration		
Description	Name of a C symbol which contains the configuration of the underlying cryptographic primitive.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants

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

SWS Item	**ECUC_Csm_00771 :**		
Name	CsmKeyDeriveSymKeyPrimitiveName		
Description	Name of the cryptographic primitive to use.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
Post-Build Variant Value	false		
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.*