# MICROSAR CSM

Technical Reference

Version 1.5

| | |
|---|---|
| Authors | Markus Schneider, Philipp Ritter |
| Status | Released |

# Document Information

## History

| Author | Date | Version | Remarks |
|---|---|---|---|
| Philipp Ritter | 2012-10-01 | 1.00 | Initial Version of MICROSAR Csm |
| Markus Schneider | 2013-09-24 | 1.01 | Adapted Configuration Chapter |
| Markus Schneider | 2014-02-06 | 1.02 | Adapted Service Port Chapter |
| Markus Schneider | 2015-08-27 | 1.03 | Corrections due to SafeBSW process |
| Markus Schneider | 2015-11-18 | 1.04 | Minor corrections |
| Markus Schneider | 2016-02-24 | 1.05 | Minor corrections |

## Reference Documents

| No. | Source | Title | Version |
|---|---|---|---|
| [1] | AUTOSAR | AUTOSAR_SWS_CryptoServiceManager.pdf | 1.2.0 |
| [2] | AUTOSAR | AUTOSAR_SWS_DevelopmentErrorTracer.pdf | 3.2.0 |
| [3] | AUTOSAR | AUTOSAR_SWS_DiagnosticEventManager.pdf | 4.2.0 |
| [4] | AUTOSAR | AUTOSAR_TR_BSWModuleList.pdf | 1.6.0 |
| [5] | AUTOSAR | AUTOSAR_SWS_RTE.pdf | 3.2.0 |

## Contents

## Illustrations

## Tables

# 1. Component History

The component history gives an overview over the important milestones that are supported in the different versions of the component.

| Component Version | New Features |
|---|---|
| 1.00.00 | Initial version |
| 2.00.00 | DaVinci Configurator 5 support added |
| 2.02.00 | SafeBSW |

Table 1-1 Component history

# 2. Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CSM as specified in [1].

| | | |
|---|---|---|
| **Supported AUTOSAR Release*:** | 4 | |
| **Supported Configuration Variants:** | pre-compile | |
| **Vendor ID:** | CSM_VENDOR_ID | 30 decimal<br>(= Vector-Informatik, according to HIS) |
| **Module ID:** | CSM_MODULE_ID | 110 decimal<br>(according to ref. [4]) |

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

The Crypto Service Manager (CSM) is an abstraction layer to offer a unique access to underlying basic cryptographic functionalities. Therefore, synchronous or asynchronous services are provided for which several configurations may exist.

## 2.1 Architecture Overview

The following figure shows where the CSM is located in the AUTOSAR architecture.

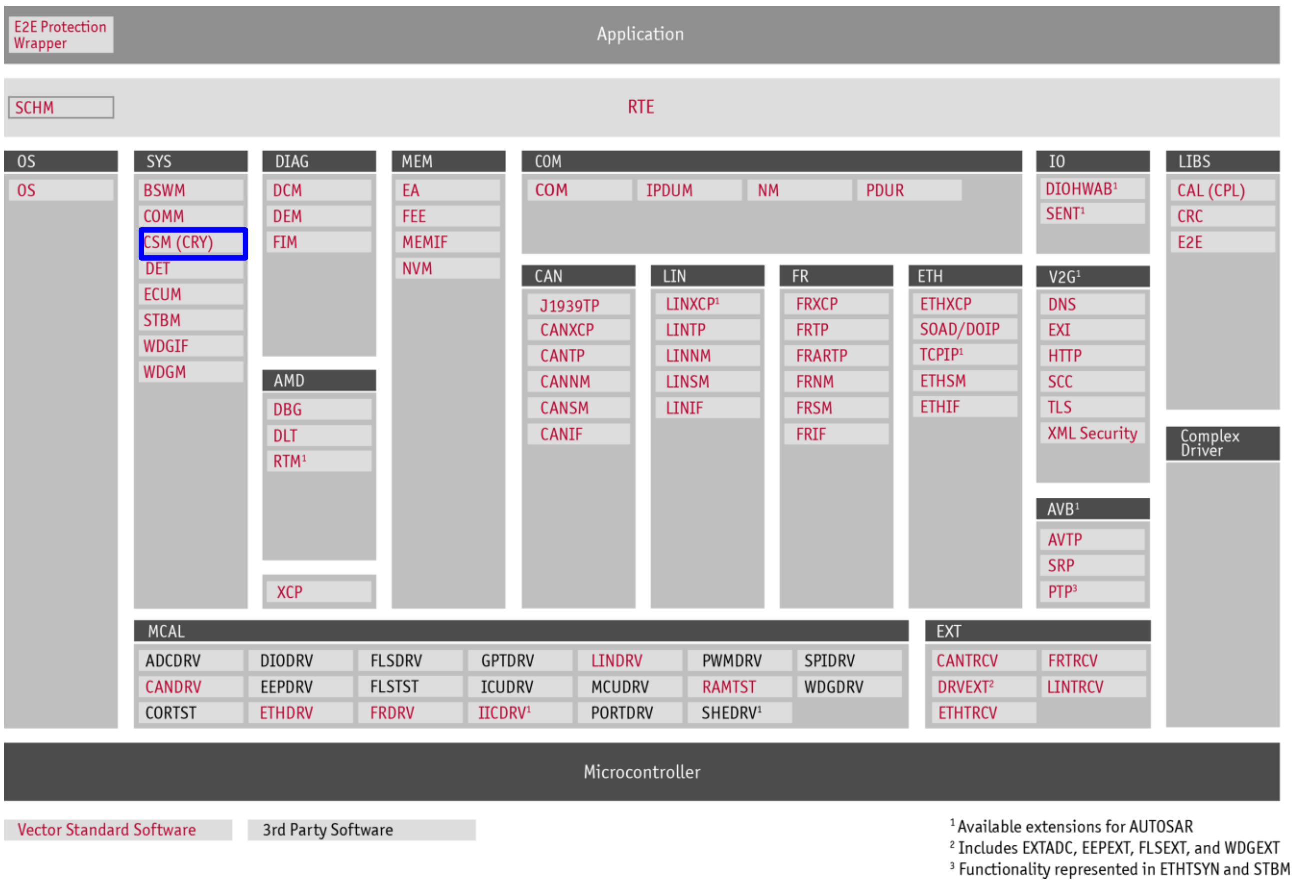

Figure 2-1 AUTOSAR 4.x Architecture Overview

Figure 2-2 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the CSM. These interfaces are described in chapter 5.

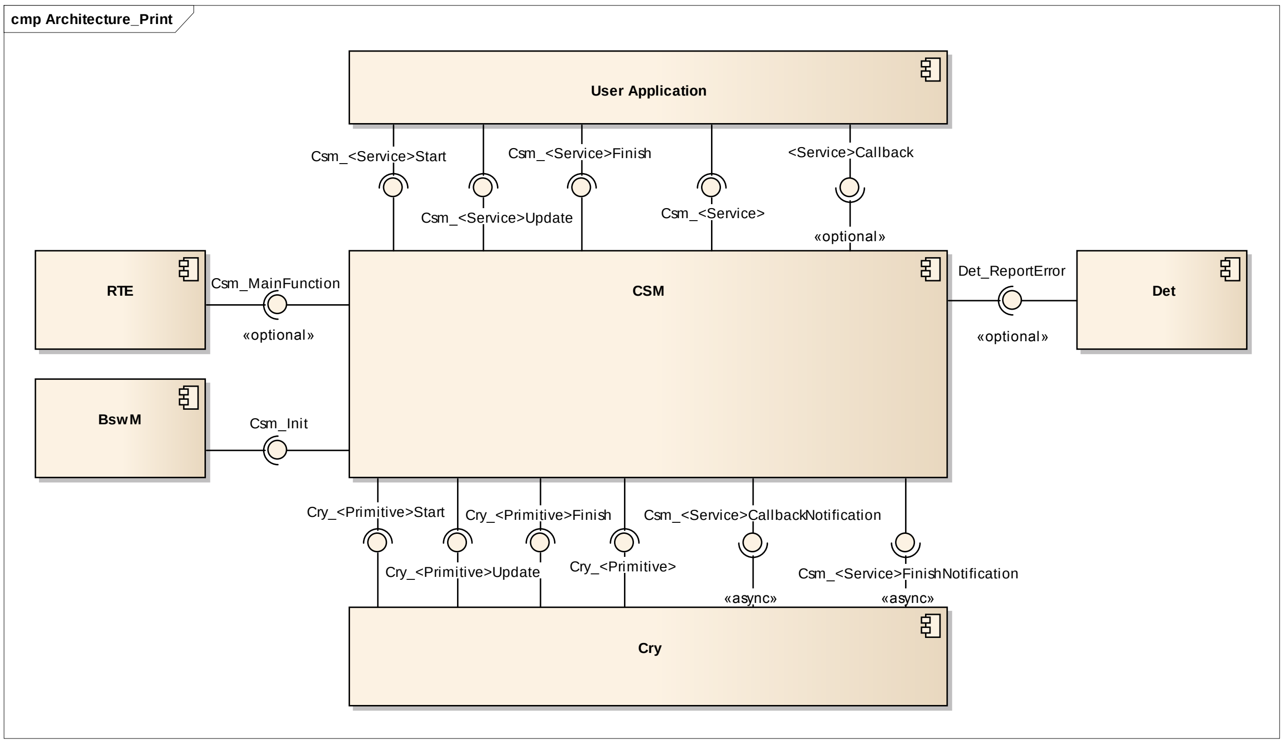

Figure 2-3 Interfaces to adjacent modules of the CSM

# 3. Functional Description

## 3.1 Features

The features listed in the following tables cover the complete functionality specified for the CSM.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

For further information of not supported features see also chapter 7.

Vector Informatik provides further CSM functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

| Supported AUTOSAR Standard Conform Features |
|---|
| All mentioned services are supported (5.2) |
| Synchronous job processing |
| Asynchronous job processing |
| Development Error Detection |
| Debugging Concept |
| Configuration through BSWMD with DaVinci Configurator Pro 5 |
| Ports and Port Interfaces (RTE Support) |

Table 3-1 Supported AUTOSAR standard conform features

The following features specified in [1] are not supported:

| Not Supported AUTOSAR Standard Conform Features |
|---|
| Interruption of job processing |
| No support of DEM |

Table 3-2 Not supported AUTOSAR standard conform features

The following features are provided beyond the AUTOSAR standard:

| Features Provided Beyond The AUTOSAR Standard |
|---|
| Unused service APIs can be deactivated |

Table 3-3 Features provided beyond the AUTOSAR standard

## 3.2 Initialization

Before calling any other functionality of the CSM module the initialization function `Csm_Init()` has to be called by the application. The initialization call shall take place after initializing the corresponding cryptographic modules.

For API details refer to chapter 5.2.1 ‘Csm_Init’.

The CSM module assumes that some variables are initialized with certain values at start-up. As not all embedded targets support the initialization of RAM within the start-up code the CSM module provides the function `Csm_InitMemory()`. This function has to be called during start-up and before Csm_Init() is called. Refer also to chapter 7.3 ‘Memory Initialization’.

For API details refer to chapter 5.2.2 ‘Csm_InitMemory’.

## 3.3 States

The CSM module stores a state for every service which clarifies if a service is active or idle. The service state is set to active in the Csm_<Service>Start function if the return value is CSM_E_OK. To reset a state to idle, e.g. due to service cancelation during update process, the specific Csm_<Service>Finish function has to be called.

## 3.4 Main Functions

The CSM module implementation provides one main function. When the usage of asynchronous job processing is enabled, this main function has to be called cyclically on task level. The default cycle time is 10 milliseconds. The main function is responsible to execute active services by calling the main function of the corresponding cryptographic primitive.

For API details refer to chapter 5.2.3 ‘Csm_MainFunction’.

## 3.5 Asynchronous Handling

There are some differences in the handling between asynchronous and synchronous mode. Asynchronous services need external state machines in the application to track the progress. When calling Csm_<Service>Start() the specific CRY function is called. The function stores the provided pointer and data provided by the API internally. Processing of data is triggered in the specific Cry_<ServiceName>MainFunction(). The configured user callback function indicates that the processing is finished carrying the result of the operation. Depending on the result, the next operation can be performed e.g. Csm_<Service>Update(). Figure 3-1 depicts this sequence.

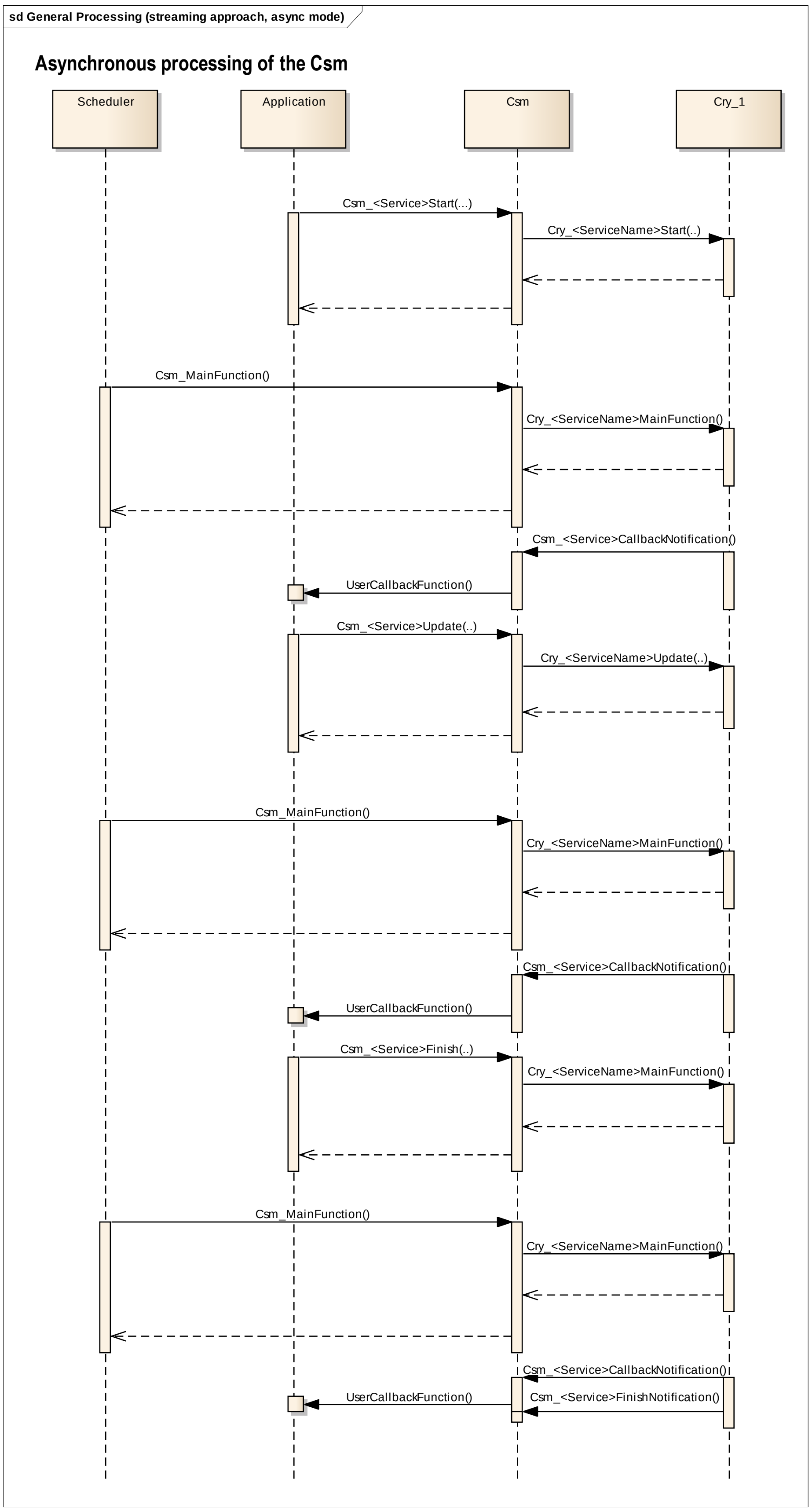
sd General Processing (streaming approach, async mode)
Asynchronous processing of the Csm
Scheduler
Application
Csm
Cry_1
Csm_<Service>Start(...)
Cry_<ServiceName>Start(..)
Csm_MainFunction()
Cry_<ServiceName>MainFunction()
Csm_<Service>CallbackNotification()
UserCallbackFunction()
Csm_<Service>Update(..)
Cry_<ServiceName>Update(..)
Csm_MainFunction()
Cry_<ServiceName>MainFunction()
Csm_<Service>CallbackNotification()
UserCallbackFunction()
Csm_<Service>Finish(..)
Cry_<ServiceName>MainFunction()
Csm_MainFunction()
Cry_<ServiceName>MainFunction()
Csm_<Service>CallbackNotification()
UserCallbackFunction()
Csm_<Service>FinishNotification()

Figure 3-1 CSM asynchronous mode

**Caution**
All input and output data buffers have to be valid during the whole processing, not only for the execution of the service call itself.

## 3.6 Error Handling

### 3.6.1 Development Error Reporting

If development error reporting is enabled (i.e. pre-compile parameter `CSM_DEV_ERROR_REPORT == STD_ON`), reporting of development errors is done by the service

```
    Std_ReturnType Det_ReportError (
            uint16 ModuleId, uint8 InstanceId,
            uint8 ApiId, uint8 ErrorId )                    (5.3)
```

Please refer to the documentation of the development error tracer [2] for further information and a detailed description of the API.

The reported CSM ID is 110.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

| Service ID | | Service |
|---|---|---|
| 0x03 | CSM_HASHSTART_ID | Csm_HashStart() |
| 0x04 | CSM_HASHUPDATE_ID | Csm_HashUpdate() |
| 0x05 | CSM_HASHFINISH_ID | Csm_HashFinish() |
| 0x06 | CSM_MACGENERATESTART_ID | Csm_MacGenerateStart() |
| 0x07 | CSM_MACGENERATEUPDATE_ID | Csm_MacGenerateUpdate() |
| 0x08 | CSM_MACGENERATEFINISH_ID | Csm_MacGenerateFinish() |
| 0x09 | CSM_MACVERIFYSTART_ID | Csm_MacVerifyStart() |
| 0x0A | CSM_MACVERIFYUPDATE_ID | Csm_MacVerifyUpdate() |
| 0x0B | CSM_MACVERIFYFINISH_ID | Csm_MacVerifyFinish() |
| 0x0C | CSM_RANDOMSEEDSTART_ID | Csm_RandomSeedStart() |
| 0x0D | CSM_RANDOMSEEDUPDATE_ID | Csm_RandomSeedUpdate() |
| 0x0E | CSM_RANDOMSEEDFINISH_ID | Csm_RandomSeedFinish() |
| 0x0F | CSM_RANDOMGENERATE_ID | Csm_RandomGenerate() |
| 0x10 | CSM_SYMBLOCKENCRYPTSTART_ID | Csm_SymBlockEncryptStart() |
| 0x11 | CSM_SYMBLOCKENCRYPTUPDATE_ID | Csm_SymBlockEncryptUpdate() |
| 0x12 | CSM_SYMBLOCKENCRYPTFINISH_ID | Csm_SymBlockEncryptFinish() |
| 0x13 | CSM_SYMBLOCKDECRYPTSTART_ID | Csm_SymBlockDecryptStart() |
| 0x14 | CSM_SYMBLOCKDECRYPTUPDATE_ID | Csm_SymBlockDecryptUpdate() |
| 0x15 | CSM_SYMBLOCKDECRYPTFINISH_ID | Csm_SymBlockDecryptFinish() |
| 0x16 | CSM_SYMENCRYPTSTART_ID | Csm_SymEncryptStart() |
| 0x17 | CSM_SYMENCRYPTUPDATE_ID | Csm_SymEncryptUpdate() |
| 0x18 | CSM_SYMENCRYPTFINISH_ID | Csm_SymEncryptFinish() |

| Service ID | | Service |
|---|---|---|
| 0x19 | CSM_SYMDECRYPTSTART_ID | Csm_SymDecryptStart() |
| 0x1A | CSM_SYMDECRYPTUPDATE_ID | Csm_SymDecryptUpdate() |
| 0x1B | CSM_SYMDECRYPTFINISH_ID | Csm_SymDecryptFinish() |
| 0x1C | CSM_ASYMENCRYPTSTART_ID | Csm_AsymEncryptStart() |
| 0x1D | CSM_ASYMENCRYPTUPDATE_ID | Csm_AsymEncryptUpdate() |
| 0x1E | CSM_ASYMENCRYPTFINISH_ID | Csm_AsymEncryptFinish() |
| 0x1F | CSM_ASYMDECRYPTSTART_ID | Csm_AsymDecryptStart() |
| 0x20 | CSM_ASYMDECRYPTUPDATE_ID | Csm_AsymDecryptUpdate() |
| 0x21 | CSM_ASYMDECRYPTFINISH_ID | Csm_AsymDecryptFinish() |
| 0x22 | CSM_SIGNATUREGENERATESTART_ID | Csm_SignatureGenerateStart() |
| 0x23 | CSM_SIGNATUREGENERATEUPDATE_ID | Csm_SignatureGenerateUpdate() |
| 0x24 | CSM_SIGNATUREGENERATEFINISH_ID | Csm_SignatureGenerateFinish() |
| 0x25 | CSM_SIGNATUREVERIFYSTART_ID | Csm_SignatureVerifyStart() |
| 0x26 | CSM_SIGNATUREVERIFYUPDATE_ID | Csm_SignatureVerifyUpdate() |
| 0x27 | CSM_SIGNATUREVERIFYFINISH_ID | Csm_SignatureVerifyFinish() |
| 0x28 | CSM_CHECKSUMSTART_ID | Csm_ChecksumStart() |
| 0x29 | CSM_CHECKSUMUPDATE_ID | Csm_ChecksumUpdate() |
| 0x2A | CSM_CHECKSUMFINISH_ID | Csm_ChecksumFinish() |
| 0x2B | CSM_KEYDERIVESTART_ID | Csm_KeyDeriveStart() |
| 0x2C | CSM_KEYDERIVEUPDATE_ID | Csm_KeyDeriveUpdate() |
| 0x2D | CSM_KEYDERIVEFINISH_ID | Csm_KeyDeriveFinish() |
| 0x4C | CSM_KEYDERIVESYMKEY_ID | Csm_KeyDeriveSymKey() |
| 0x2E | CSM_KEYEXCHANGECALCPUBVAL_ID | Csm_KeyExchangeCalcPubVal() |
| 0x2F | CSM_KEYEXCHANGECALCSECRETSTART_ID | Csm_KeyExchangeCalcSecretStart() |
| 0x30 | CSM_KEYEXCHANGECALCSECRETUPDATE_ID | Csm_KeyExchangeCalcSecretUpdate() |
| 0x31 | CSM_KEYEXCHANGECALCSECRETFINISH_ID | Csm_KeyExchangeCalcSecretFinish() |
| 0x3D | CSM_KEYEXCHANGECALCSYMKEYSTART_ID | Csm_KeyExchangeCalcSymKeyStart() |
| 0x3E | CSM_KEYEXCHANGECALCSYMKEYUPDATE_ID | Csm_KeyExchangeCalcSymKeyUpdate() |
| 0x3F | CSM_KEYEXCHANGECALCSYMKEYFINISH_ID | Csm_KeyExchangeCalcSymKeyFinish() |
| 0x32 | CSM_SYMKEYEXTRACTSTART_ID | Csm_SymKeyExtractStart() |
| 0x33 | CSM_SYMKEYEXTRACTUPDATE_ID | Csm_SymKeyExtractUpdate() |
| 0x34 | CSM_SYMKEYEXTRACTFINISH_ID | Csm_SymKeyExtractFinish() |
| 0x40 | CSM_SYMKEYWRAPSYMSTART_ID | Csm_SymKeyWrapSymStart() |
| 0x41 | CSM_SYMKEYWRAPSYMUPDATE_ID | Csm_SymKeyWrapSymUpdate() |
| 0x42 | CSM_SYMKEYWRAPSYMFINISH_ID | Csm_SymKeyWrapSymFinish() |
| 0x43 | CSM_SYMKEYWRAPASYMSTART_ID | Csm_SymKeyWrapAsymStart() |
| 0x44 | CSM_SYMKEYWRAPASYMUPDATE_ID | Csm_SymKeyWrapAsymUpdate() |
| 0x45 | CSM_SYMKEYWRAPASYMFINISH_ID | Csm_SymKeyWrapAsymFinish() |
| 0x35 | CSM_ASYMPUBLICKEYEXTRACTSTART_ID | Csm_AsymPublicKeyExtractStart() |
| 0x36 | CSM_ASYMPUBLICKEYEXTRACTUPDATE_ID | Csm_AsymPublicKeyExtractUpdate() |
| 0x37 | CSM_ASYMPUBLICKEYEXTRACTFINISH_ID | Csm_AsymPublicKeyExtractFinish() |
| 0x38 | CSM_ASYMPRIVATEKEYEXTRACTSTART_ID | Csm_AsymPrivateKeyExtractStart() |
| 0x39 | CSM_ASYMPRIVATEKEYEXTRACTUPDATE_ID | Csm_AsymPrivateKeyExtractUpdate() |
| 0x3A | CSM_ASYMPRIVATEKEYEXTRACTFINISH_ID | Csm_AsymPrivateKeyExtractFinish() |
| 0x46 | CSM_ASYMPRIVATEKEYWRAPSYMSTART_ID | Csm_AsymPrivateKeyWrapSymStart() |
| 0x47 | CSM_ASYMPRIVATEKEYWRAPSYMUPDATE_ID | Csm_AsymPrivateKeyWrapSymUpdate() |
| 0x48 | CSM_ASYMPRIVATEKEYWRAPSYMFINISH_ID | Csm_AsymPrivateKeyWrapSymFinish() |

| Service ID | | Service |
|---|---|---|
| 0x49 | CSM_ASYMPRIVATEKEYWRAPASYMSTART_ID | Csm_AsymPrivateKeyWrapAsymStart() |
| 0x4A | CSM_ASYMPRIVATEKEYWRAPASYMUPDATE_ID | Csm_AsymPrivateKeyWrapAsymUpdate() |
| 0x4B | CSM_ASYMPRIVATEKEYWRAPASYMFINISH_ID | Csm_AsymPrivateKeyWrapAsymFinish() |

Table 3-4 Service IDs

The errors reported to DET are described in the following table:

| Error Code | | Description |
|---|---|---|
| 0x01 | CSM_E_PARAM_PTR_INVALID | API request called with invalid parameter (null pointer). |
| 0x02 | CSM_E_SERVICE_NOT_STARTED | Requested service is not initialized. |
| 0x03 | CSM_E_PARAM_METHOD_INVALID | API request called with invalid parameter (invalid method for selected service). |
| 0x04 | CSM_E_PARAM_KEY_TYPE_INVALID | API request called with invalid parameter (invalid key type for selected service). |
| 0x05 | CSM_E_UNINT | API request called before initialization of CSM module. |
| 0x06 | CSM_E_BUFFER_TOO_SMALL | Provided buffer for storing the result of a computation is too small. |

Table 3-5 Errors reported to DET

The following table shows which development error can occur on which services:

| Check / Service | CSM_E_PARAM_PTR_INVALID | CSM_E_SERVICE_NOT_STARTED | CSM_E_PARAM_METHOD_INVALID | CSM_E_UNINT |
|---|---|---|---|---|
| Csm_MainFunction | | | | ■ |
| Csm_<Service>Start | ■ | | ■ | ■ |
| Csm_<Service>Update | ■ | ■ | ■ | |
| Csm_<Service>Finish | ■ | ■ | ■ | |

Table 3-6 Development Error Reporting: Assignment of checks to services

### 3.6.2 Production Code Error Reporting

The current implementation of the CSM module does not report any production errors.

# 4. Integration

This chapter gives necessary information for the integration of the MICROSAR CSM into an application environment of an ECU.

## 4.1 Scope of Delivery

The delivery of the CSM contains the files which are described in the chapters 4.1.1 and 4.1.2:

### 4.1.1 Static Files

| File Name | Source Code Delivery | Object Code Delivery | Description |
|---|---|---|---|
| Csm.c | ■ | | This is the source file of the CSM |
| Csm.h | ■ | | This is the header file of the CSM. |
| Csm_Cbk.h | ■ | | This is the callback header file of the CSM |
| Csm_Types.h | ■ | | This is the type definition header file of the CSM |

Table 4-1 Static files

### 4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator Pro 5.

For more Information about the configuration see chapter 6.2 Configuration with DaVinci Configurator.

| File Name | Description |
|---|---|
| Csm_Cfg.h | This is the configuration header file. |
| Csm_Cfg.c | This is the configuration source file. |

Table 4-2 Generated files

## 4.2 Include Structure

Figure 4-1 shows the include structure of the CSM. Some includes are optional and depend on the configuration. `Cry<Primitve>.h` stands for every used cryptographic primitive.

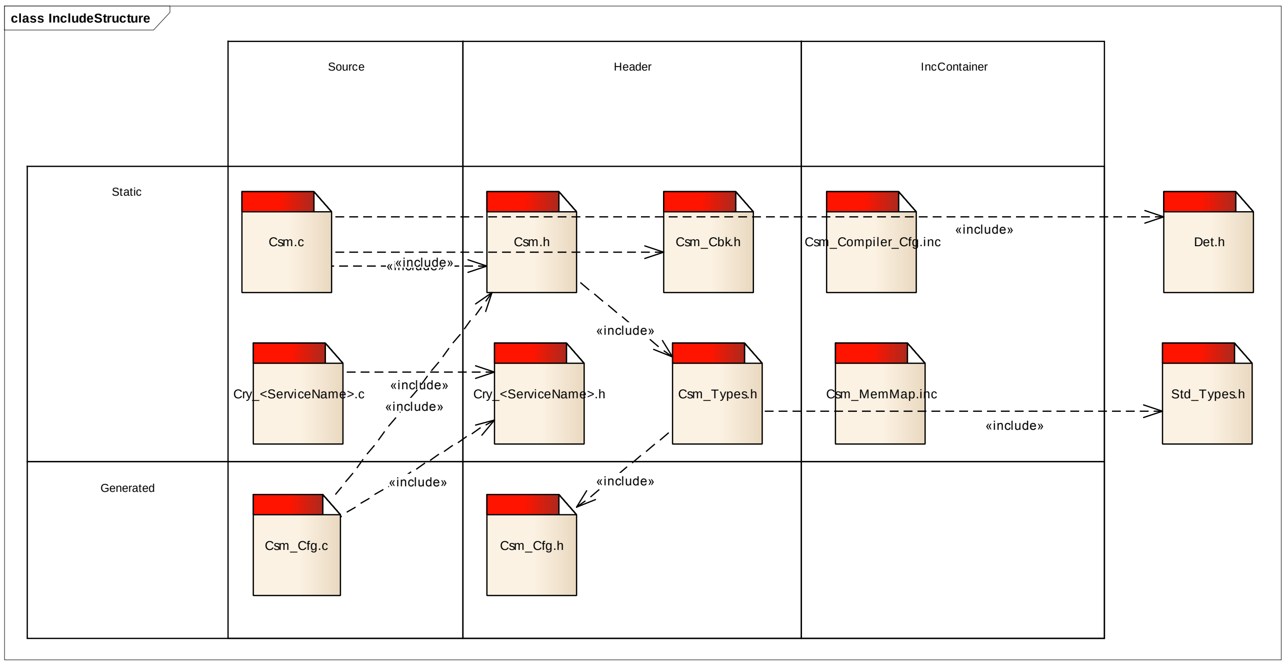

Figure 4-1 Include structure

## 4.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table (Table 4-3) contains the memory section names and the compiler abstraction definitions of the CSM and illustrates their assignment among each other.

| Memory Mapping Sections \ Compiler Abstraction Definitions | CSM_CODE | CSM_CONST | CSM_VAR_NOINIT | CSM_VAR_ZERO_INIT | CSM_APPL_VAR |
|---|---|---|---|---|---|
| CSM_START_SEC_CODE<br>CSM_STOP_SEC_CODE | ■ | | | | |
| CSM_START_SEC_CONST_8BIT<br>CSM_STOP_SEC_CONST_8BIT | | ■ | | | |
| CSM_START_SEC_CONST_UNSPECIFIED<br>CSM_STOP_SEC_CONST_UNSPECIFIED | | ■ | | | |
| CSM_START_SEC_VAR_NOINIT_8BIT<br>CSM_STOP_SEC_VAR_NOINIT_8BIT | | | ■ | | |
| CSM_START_SEC_VAR_NOINIT_16BIT<br>CSM_STOP_SEC_VAR_NOINIT_16BIT | | | ■ | | |
| CSM_START_SEC_VAR_ZERO_INIT_8BIT<br>CSM_STOP_SEC_VAR_ZERO_INIT_8BIT | | | | ■ | |

Table 4-3 Compiler abstraction and memory mapping

## 4.4 Critical Sections

The current implementation of the CSM module does not need any critical section.

# 5. API Description

For an interfaces overview please see Figure 2-3.

## 5.1 Type Definitions

The types defined by the CSM are described in this chapter.

| Type Name | C-Type | Description | Value Range |
|---|---|---|---|
| Csm_ConfigIdType | uint16 | Identification of a CSM service configuration via a numeric identifier, that is unique within a service. | 0..65535 |
| Csm_ReturnType | uint8 | Return Type of the Csm Module | `CSM_E_OK` The execution of the called function succeeded. |
| | | | `CSM_E_NOT_OK` The execution of the called function failed |
| | | | `CSM_E_BUSY` The service request failed because the service is still busy. |
| | | | `CSM_E_SMALL_BUFFER` The service request failed because the provided buffer is too small to store the result of the service. |
| | | | `CSM_E_ENTROPY_EXHAUSION` The service request failed because the entropy of the random number generator is exhausted. |
| Csm_AlignType | uint8, uint16, uint32 | A scalar type which has maximum alignment restrictions on the given platform. This value is configured by CsmMaxAlignScalarType | |
| Csm_VerifyResultType | uint8 | | `CSM_E_VER_OK` The result of the verification is "true". |
| | | | `CSM_E_VER_NOT_OK` The result of the verification is "false". |
| Csm_CallbackType* | Std_ReturnType | Function pointer for service notification callback. | |

Table 5-1 Type definitions

### Csm_AsymPublicKeyType

This structure represents a public asymmetrical key.

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| length | uint32 | This element contains the length of the key stored in element 'data' | 0..4294967295 |
| data | Csm_AlignType | This element contains the key data or a key handle. | `CSM_ASYM_PUB_KEY_MAX_SIZE` |

Table 5-2 Csm_AsymPublicKeyType

### Csm_AsymPrivateKeyType

This structure represents a private asymmetrical key.

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| length | uint32 | This element contains the length of the key stored in element 'data' | 0..4294967295 |
| data | Csm_AlignType | This element contains the key data or a key handle. | `CSM_ASYM_PUB_KEY_MAX_SIZE` |

Table 5-3 Csm_AsymPivateKeyType

### Csm_SymKeyType

This structure represents a symmetrical key.

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| length | uint32 | This element contains the length of the key stored in element 'data' | 0..4294967295 |
| data | Csm_AlignType | This element contains the key data or a key handle. | `CSM_ASYM_PRIV_KEY_MAX_SIZE` |

Table 5-4 Csm_SymKeyType

### Csm_SymKeyType

This structure represents a symmetrical key.

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| length | uint32 | This element contains the length of the key stored in element 'data' | 0..4294967295 |
| data | Csm_AlignType | This element contains the key data or a key handle. | `CSM_SYM_KEY_MAX_SIZE` |

Table 5-5 Csm_SymKeyType

### Csm_KeyExchangeBaseType

This structure represents base type information of the key exchange protocol.

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| length | uint32 | This element contains the length of the key stored in element 'data' | 0..4294967295 |
| data | Csm_AlignType | This element contains the key data or a key handle. | `CSM_KEY_EX_BASE_MAX_SIZE` |

Table 5-6 Csm_KeyExchangeBaseType

### Csm_KeyExchangePrivateType

This structure represents private information of the key exchange protocol.

| Struct Element Name | C-Type | Description | Value Range |
|---|---|---|---|
| length | uint32 | This element contains the length of the key stored in element 'data' | 0..4294967295 |
| data | Csm_AlignType | This element contains the key data or a key handle. | `CSM_KEY_EX_PRIV_MAX_SIZE` |

Table 5-7 Csm_KeyExchangePrivateType

### Csm_<Service>ConfigType

This structure is defined for each service and represents the configuration of this service.

The parameters of the several function pointers depend on the service and are nearly equal to the corresponding Csm Service function. Only the `CfgId`, which is part of every Csm service function, will be replaced by the corresponding `PrimitiveConfigPtr`.

| Struct Element Name | C-Type | Description |
|---|---|---|
| ConfigId | Csm_ConfigIdType | The numeric identifier of a configuration. |
| CallbackFct* | Csm_CallbackType | A pointer to the callback function which shall be called when the configured service has finished. This Element is only available if "CsmUseSyncJobProcessing" is disabled. |
| PrimitiveStartFct* | Csm_ReturnType | 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. |
| PrimitiveUpdateFct* | Csm_ReturnType | 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. |
| PrimitiveFinishFct* | Csm_ReturnType | 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. |
| PrimitiveFct* | Csm_ReturnType | 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. |
| PrimitiveMainFct* | void | A pointer to the function Cry_<Primitive>MainFunction of the configured cryptographic primitive. |
| PrimitiveConfigPtr* | void | A pointer to the configuration of the underlying cryptographic primitive. |

Table 5-8 Csm_<Service>ConfigType

## 5.2 Services provided by CSM

### 5.2.1 Csm_Init

| Prototype | |
|---|---|
| `void Csm_Init (void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function initializes the CSM. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function has to be called during start-up. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-9 Csm_Init

### 5.2.2 Csm_InitMemory

| Prototype | |
|---|---|
| `void Csm_InitMemory (void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| If RAM is not automatically initialized at start-up, this function must be called from start-up code to ensure that variables which must be initialized with a certain value (e.g. initialization status with UNINIT value) are set to those values. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function has to be called during start-up before the initialization is executed.<br>> This function is a Vector Extension. Refer also to chapter 7.3 ‘Memory Initialization’. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-10 Csm_InitMemory

### 5.2.3 Csm_MainFunction

| **Prototype** | |
|---|---|
| `void Csm_MainFunction (void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function implements the asynchronous service handling.<br>**Note**<br>This function is empty if 'Use Sync Job Processing' is enabled. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is not reentrant.<br>> This function has to be called cyclically on task level by BSW Scheduler.<br>> This function must not be called by the application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-11 Csm_MainFunction

### 5.2.4 Csm_Interruption

| **Prototype** | |
|---|---|
| `void Csm_Interruption (void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function has no functionality and exists only for compatibility reasons. | |
| **Particularities and Limitations** | |
| > This function has no functionality. | |
| Call Context | |
| > This function can be called from task and interrupt level. | |

Table 5-12 Csm_Interruption

## 5.2.5 Csm_GetVersionInfo

| Prototype | |
|---|---|
| `void` **`Csm_GetVersionInfo`** `(Std_VersionInfoType *csmVerInfoPtr)` | |
| **Parameter** | |
| `csmVerInfoPtr` | Pointer where the version information shall be copied to. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function copies the CSM version information to the location provided by the pointer. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is only available if 'Version Info Api" is enabled. | |
| Call Context | |
| > This function can be called from task and interrupt level. | |

Table 5-13 Csm_GetVersionInfo

## 5.2.6 Csm_HashStart

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_HashStart`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the hash computation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-14 Csm_HashStart

### 5.2.7 Csm_HashUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_HashUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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 hash value shall be computed. |
| dataLength | Contains the number of bytes for which the hash value shall be computed. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the hash computation service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-15 Csm_HashUpdate

### 5.2.8 Csm_HashFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_HashFinish`** `(Csm_ConfigIdType cfgId, uint8 *resultPtr, uint32 *resultLengthPtr, boolean truncationIsAllowed)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| resultPtr | Holds a pointer to the memory location which will hold the hash value. If the hash value does not fit into the given buffer, and truncation is allowed, the result shall be truncated. |
| 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 provided buffer. When the request has finished, the actual length of the returned hash value shall be stored. |
| truncationIsAllowed | This parameter states whether a truncation of the result is allowed or not. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the hash computation service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-16 Csm_HashFinish

### 5.2.9 Csm_MacGenerateStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_MacGenerateStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the MAC generation operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the MAC generation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-17 Csm_MacGenerateStart

### 5.2.10 Csm_MacGenerateUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_MacGenerateUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the MAC generation service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-18 Csm_MacGenerateUpdate

### 5.2.11 Csm_MacGenerateFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_MacGenerateFinish`** `(Csm_ConfigIdType cfgId, uint8 *resultPtr, uint32 *resultLengthPtr, boolean truncationIsAllowed)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| resultPtr | Holds a pointer to the memory location which will hold the MAC. If the MAC does not fit into the given buffer, and truncation is allowed, the result shall be truncated. |
| 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 provided buffer. When the request has finished, the actual length of the returned MAC shall be stored. |
| truncationIsAllowed | This parameter states whether a truncation of the result is allowed or not. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the MAC generation service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-19 Csm_MacGenerateFinish

### 5.2.12 Csm_MacVerifyStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_MacVerifyStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the MAC verification operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the MAC verification service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-20 Csm_MacVerifyStart

### 5.2.13 Csm_MacVerifyUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_MacVerifyUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the MAC verification service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-21 Csm_MacVerifyUpdate

### 5.2.14 Csm_MacVerifyFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_MacVerifyFinish`** `(Csm_ConfigIdType cfgId, const uint8 *MacPtr, uint32 MacLength, Csm_VerifyResultType *resultPtr)` | |
| **Parameter** | |
| 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 |
| resultPtr | Holds a pointer to the memory location which will hold the result of the verification. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the MAC verification service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-22 Csm_MacVerifyFinish

### 5.2.15 Csm_RandomSeedStart

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_RandomSeedStart`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the random seed service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-23 Csm_RandomSeedStart

### 5.2.16 Csm_RandomSeedUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_RandomSeedUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *seedPtr, uint32 seedLength)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| seedPtr | Holds a pointer to a source of entropy which is used to provide a seed for the random number generator. |
| seedLength | Contains the number of bytes for which the seed shall be computed. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the random seed service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-24 Csm_RandomSeedUpdate

### 5.2.17 Csm_RandomSeedFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType **Csm_RandomSeedFinish** (Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the random seed service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-25 Csm_RandomSeedFinish

### 5.2.18 Csm_RandomGenerate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_RandomGenerate`** `(Csm_ConfigIdType cfgId, uint8 *resultPtr, uint32 resultLength)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| resultPtr | Holds a pointer to the memory location which will hold the random number. |
| resultLength | Contains the number of bytes for which the random number shall be computed. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the random number generation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-26 Csm_RandomGenerate

### 5.2.19 Csm_SymBlockEncryptStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymBlockEncryptStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the symmetrical block encryption operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical block encryption service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-27 Csm_SymBlockEncryptStart

### 5.2.20 Csm_SymBlockEncryptUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymBlockEncryptUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *plainTextPtr, uint32 plainTextLength, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| plainTextPtr | Holds a pointer to the data for which a encrypted text shall be computed. |
| plainTextLength | Contains the number of bytes for which the encrypted text shall be computed. |
| cipherTextPtr | Holds a pointer to the memory location which will hold the encrypted text. |
| cipherTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical block encryption service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-28 Csm_SymBlockEncryptUpdate

### 5.2.21 Csm_SymBlockEncryptFinish

| Prototype | |
|---|---|
| Csm_ReturnType **Csm_SymBlockEncryptFinish** (Csm_ConfigIdType cfgId) | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical block encryption service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-29 Csm_SymBlockEncryptFinish

### 5.2.22 Csm_SymBlockDecryptStart

| Prototype | |
|---|---|
| Csm_ReturnType **Csm_SymBlockDecryptStart** (Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr) | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the symmetrical block decryption operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical block decryption service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |

| Call Context |
|---|
| > This function can be called from task level only. |

Table 5-30 Csm_SymBlockDecryptStart

## 5.2.23 Csm_SymBlockDecryptUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_SymBlockDecryptUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *cipherTextPtr, uint32 cipherTextLength, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| cipherTextPtr | Holds a pointer to the data for which a decrypted text shall be computed. |
| cipherTextLength | Contains the number of bytes for which the decrypted text shall be computed. |
| plainTextPtr | Holds a pointer to the memory location which will hold the decrypted text. |
| plainTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical block decryption service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-31 Csm_SymBlockDecryptUpdate

### 5.2.24 Csm_SymBlockDecryptFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymBlockDecryptFinish`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical block decryption service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-32 Csm_SymBlockDecryptFinish

### 5.2.25 Csm_SymEncryptStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymEncryptStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr, const uint8 *InitVectorPtr, uint32 InitVectorLength)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the symmetrical encryption operation. |
| InitVectorPtr | Holds a pointer to the initialisation vector which has to be used. |
| InitVectorLength | Contains the number of bytes provided as the initialisation vector. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical encryption service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-33 Csm_SymEncryptStart

### 5.2.26 Csm_SymEncryptUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymEncryptUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *plainTextPtr, uint32 plainTextLength, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| plainTextPtr | Holds a pointer to the data for which a encrypted text shall be computed. |
| plainTextLength | Contains the number of bytes for which the encrypted text shall be computed. |
| cipherTextPtr | Holds a pointer to the memory location which will hold the encrypted text. |
| cipherTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical encryption service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-34 Csm_SymEncryptUpdate

### 5.2.27 Csm_SymEncryptFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymEncryptFinish`** `(Csm_ConfigIdType cfgId, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| cipherTextPtr | Holds a pointer to the memory location which will hold the encrypted text. |
| cipherTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical encryption service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-35 Csm_SymEncryptFinish

### 5.2.28 Csm_SymDecryptStart

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_SymDecryptStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr, const uint8 *InitVectorPtr, uint32 InitVectorLength)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the symmetrical decryption operation. |
| InitVectorPtr | Holds a pointer to initialisation vector which has to be used during the symmetrical decryption. |
| InitVectorLength | Holds a pointer to the initialisation vector which has to be used during the symmetrical decryption. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical decryption service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-36 Csm_SymDecryptStart

### 5.2.29 Csm_SymDecryptUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymDecryptUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *cipherTextPtr, uint32 cipherTextLength, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| cipherTextPtr | Holds a pointer to the data for which a decrypted text shall be computed. |
| cipherTextLength | Contains the number of bytes for which the decrypted text shall be computed. |
| plainTextPtr | Holds a pointer to the memory location which will hold the decrypted text. |
| plainTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical decryption service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-37 Csm_SymDecryptUpdate

### 5.2.30 Csm_SymDecryptFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymDecryptFinish`** `(Csm_ConfigIdType cfgId, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| plainTextPtr | Holds a pointer to the memory location which will hold the decrypted text. |
| plainTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical decryption service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-38 Csm_SymDecryptFinish

### 5.2.31 Csm_AsymEncryptStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymEncryptStart`** `(Csm_ConfigIdType cfgId, const Csm_AsymPublicKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the asymmetrical encryption operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the asymmetrical encryption service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-39 Csm_AsymEncryptStart

### 5.2.32 Csm_AsymEncryptUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymEncryptUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *plainTextPtr, uint32 plainTextLength, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| plainTextPtr | Holds a pointer to the data for which a encrypted text shall be computed. |
| plainTextLength | Contains the number of bytes for which the encrypted text shall be computed. |
| cipherTextPtr | Holds a pointer to the memory location which will hold the encrypted text. |
| cipherTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to feed the asymmetrical encryption service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-40 Csm_AsymEncryptUpdate

### 5.2.33 Csm_AsymEncryptFinish

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymEncryptFinish`** `(Csm_ConfigIdType cfgId, uint8 *cipherTextPtr, uint32 *cipherTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| cipherTextPtr | Holds a pointer to the memory location which will hold the encrypted text. |
| cipherTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned encrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the asymmetrical encryption service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-41 Csm_AsymEncryptFinish

### 5.2.34 Csm_AsymDecryptStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymDecryptStart`** `(Csm_ConfigIdType cfgId, const Csm_AsymPrivateKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the asymmetrical decryption operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the asymmetrical decryption service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-42 Csm_AsymDecryptStart

### 5.2.35 Csm_AsymDecryptUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymDecryptUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *cipherTextPtr, uint32 cipherTextLengthPtr, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| cipherTextPtr | Holds a pointer to the data for which a decrypted text shall be computed. |
| cipherTextLengthPtr | Contains the number of bytes for which the decrypted text shall be computed. |
| plainTextPtr | Holds a pointer to the memory location which will hold the decrypted text. |
| plainTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to feed the asymmetrical decryption service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-43 Csm_AsymDecryptUpdate

### 5.2.36 Csm_AsymDecryptFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymDecryptFinish`** `(Csm_ConfigIdType cfgId, uint8 *plainTextPtr, uint32 *plainTextLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| plainTextPtr | Holds a pointer to the memory location which will hold the decrypted text. |
| plainTextLengthPtr | 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 provided buffer. When the request has finished, the actual length of the returned decrypted text shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the asymmetrical decryption service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-44 Csm_AsymDecryptFinish

### 5.2.37 Csm_SignatureGenerateStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SignatureGenerateStart`** `(Csm_ConfigIdType cfgId, const Csm_AsymPrivateKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the signature generate operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the signature generate service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-45 Csm_SignatureGenerateStart

### 5.2.38 Csm_SignatureGenerateUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SignatureGenerateUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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 signature shall be computed. |
| dataLength | Contains the number of bytes for which the signature shall be computed. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the signature generate service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-46 Csm_SignatureGenerateUpdate

### 5.2.39 Csm_SignatureGenerateFinish

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_SignatureGenerateFinish`** `(Csm_ConfigIdType cfgId, uint8 *resultPtr, uint32 *resultLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| resultPtr | Holds a pointer to the memory location which will hold the signature. |
| 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 provided buffer. When the request has finished, the actual length of the returned signature shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the signature generate service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-47 Csm_SignatureGenerateFinish

### 5.2.40 Csm_SignatureVerifyStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SignatureVerifyStart`** `(Csm_ConfigIdType cfgId, const Csm_AsymPublicKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the signature verification operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the signature verification service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-48 Csm_SignatureVerifyStart

### 5.2.41 Csm_SignatureVerifyUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SignatureVerifyUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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 signature shall be computed. |
| dataLength | Contains the number of bytes for which the signature shall be computed. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the signature verification service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-49 Csm_SignatureVerifyUpdate

### 5.2.42 Csm_SignatureVerifyFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SignatureVerifyFinish`** `(Csm_ConfigIdType cfgId, const uint8 *signaturePtr, uint32 signatureLength, Csm_VerifyResultType *resultPtr)` | |
| **Parameter** | |
| 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 |
| resultPtr | Holds a pointer to the memory location which will hold the result of the signature verification. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the signature verification service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-50 Csm_SignatureVerifyFinish

### 5.2.43 Csm_ChecksumStart

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_ChecksumStart`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the checksum generation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-51 Csm_ChecksumStart

### 5.2.44 Csm_ChecksumUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_ChecksumUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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 checksum shall be computed. |
| dataLength | Contains the number of bytes for which the checksum shall be computed. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the checksum generation service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |

| Call Context |
|---|
| > This function can be called from task level only. |

Table 5-52 Csm_ChecksumUpdate

### 5.2.45 Csm_ChecksumFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_ChecksumFinish`** `(Csm_ConfigIdType cfgId, uint8 *resultPtr, uint32 *resultLengthPtr, boolean truncationIsAllowed)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| resultPtr | Holds a pointer to the memory location which will hold the checksum. If the checksum does not fit into the given buffer, and truncation is allowed, the result shall be truncated. |
| 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 provided buffer. When the request has finished, the actual length of the returned checksum shall be stored. |
| truncationIsAllowed | This parameter states whether a truncation of the result is allowed or not. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the checksum generation service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-53 Csm_ChecksumFinish

### 5.2.46 Csm_KeyDeriveStart

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyDeriveStart`** `(Csm_ConfigIdType cfgId, uint32 keyLength, uint32 iterations)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the Key Derivation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-54 Csm_KeyDeriveStart

### 5.2.47 Csm_KeyDeriveUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyDeriveUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *passwordPtr, uint32 passwordLength, const uint8 *saltPtr, uint32 saltLength)` | |
| **Parameter** | |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the Key Derivation service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-55 Csm_KeyDeriveUpdate

### 5.2.48 Csm_KeyDeriveFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyDeriveFinish`** `(Csm_ConfigIdType cfgId, Csm_SymKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the memory location which will hold the derived key. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the Key Derivation service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-56 Csm_KeyDeriveFinish

### 5.2.49 Csm_KeyDeriveSymKey

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyDeriveSymKey`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *baseKeyPtr, const uint8 *customisationValPtr, uint32 customisationValLength, Csm_SymKeyType *derivedKeyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| derivedKeyPtr | Holds a pointer to the memory location which will hold the result of the key derivation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the Key Derivation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-57 Csm_KeyDeriveSymKey

### 5.2.50 Csm_KeyExchangeCalcPubVal

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcPubVal`** `(Csm_ConfigIdType cfgId, const Csm_KeyExchangeBaseType *basePtr, const Csm_KeyExchangePrivateType *privateValuePtr, uint8 *publicValuePtr, uint32 *publicValueLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| publicValuePtr | Holds a pointer to the memory location which will hold the public value. |
| publicValueLengthPtr | Holds a pointer to the number of bytes for the input buffer and the number of actual written bytes if the request was successful. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to initialize the public value calculation service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-58 Csm_KeyExchangeCalcPubVal

### 5.2.51 Csm_KeyExchangeCalcSecretStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcSecretStart`** `(Csm_ConfigIdType cfgId, const Csm_KeyExchangeBaseType *basePtr, const Csm_KeyExchangePrivateType *privateValuePtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the Key Exchange service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-59 Csm_KeyExchangeCalcSecretStart

### 5.2.52 Csm_KeyExchangeCalcSecretUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcSecretUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *partnerPublicValuePtr, uint32 partnerPublicValueLength)` | |
| **Parameter** | |
| 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 | Holds the length of the part of the partner value in bytes. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the Key Exchange service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-60 Csm_KeyExchangeCalcSecretUpdate

### 5.2.53 Csm_KeyExchangeCalcSecretFinish

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcSecretFinish`** `(Csm_ConfigIdType cfgId, uint8 *sharedSecretPtr, uint32 *sharedSecretLengthPtr, boolean truncationIsAllowed)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| sharedSecretPtr | Holds a pointer to the memory location which will hold the secret key. If the secret key does not fit into the given buffer, and truncation is allowed, the result shall be truncated. |
| sharedSecretLengthPtr | Holds a pointer to the number of bytes for which a secret key shall be computed. |
| truncationIsAllowed | This parameter states whether a truncation of the result is allowed or not. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_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. |
| **Functional Description** | |
| This interface shall be used to finish the Key Exchange service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-61 Csm_KeyExchangeCalcSecretFinish

### 5.2.54 Csm_KeyExchangeCalcSymKeyStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcSymKeyStart`** `(Csm_ConfigIdType cfgId, const Csm_KeyExchangeBaseType *basePtr, const Csm_KeyExchangePrivateType *privateValuePtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the key exchange service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-62 Csm_KeyExchangeCalcSymKeyStart

### 5.2.55 Csm_KeyExchangeCalcSymKeyUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcSymKeyUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *partnerPublicValuePtr, uint32 partnerPublicValueLength)` | |
| **Parameter** | |
| 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 | Holds the length of the part of the partner value in bytes. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the key exchange service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-63 Csm_KeyExchangeCalcSymKeyUpdate

### 5.2.56 Csm_KeyExchangeCalcSymKeyFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_KeyExchangeCalcSymKeyFinish`** `(Csm_ConfigIdType cfgId, Csm_SymKeyType *sharedKeyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| sharedKeyPtr | Holds a pointer to the memory location which will hold the shared key. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the key exchange service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-64 Csm_KeyExchangeCalcSymKeyFinish

### 5.2.57 Csm_SymKeyExtractStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType Csm_SymKeyExtractStart (Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the key which has to be used during the symmetrical key extraction operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical key extraction service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-65 Csm_SymKeyExtractStart

### 5.2.58 Csm_SymKeyExtractUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyExtractUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical key extraction service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-66 Csm_SymKeyExtractUpdate

### 5.2.59 Csm_SymKeyExtractFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyExtractFinish`** `(Csm_ConfigIdType cfgId, Csm_SymKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to a structure where the result (i.e. the symmetrical key) is stored in. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical key extraction service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-67 Csm_SymKeyExtractFinish

### 5.2.60 Csm_SymKeyWrapSymStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyWrapSymStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr, const Csm_SymKeyType *wrappingkeyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the symmetric key to be wrapped. |
| wrappingkeyPtr | Holds a pointer to the key used for wrapping. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical key wrapping service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-68 Csm_SymKeyWrapSymStart

### 5.2.61 Csm_SymKeyWrapSymUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyWrapSymUpdate`** `(Csm_ConfigIdType cfgId, uint8 *dataPtr, uint32 *dataLength)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| dataLength | 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 provided buffer. When the request has finished, the actual length of the returned wrapped key shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical key wrapping service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-69 Csm_SymKeyWrapSymUpdate

### 5.2.62 Csm_SymKeyWrapSymFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyWrapSymFinish`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical key wrapping service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-70 Csm_SymKeyWrapSymFinish

### 5.2.63 Csm_SymKeyWrapAsymStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyWrapAsymStart`** `(Csm_ConfigIdType cfgId, const Csm_SymKeyType *keyPtr, const Csm_AsymPublicKeyType *wrappingkeyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the symmetric key to be wrapped. |
| wrappingkeyPtr | Holds a pointer to the key used for wrapping. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the symmetrical key wrapping service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |

| Call Context |
|---|
| > This function can be called from task level only. |

Table 5-71 Csm_SymKeyWrapAsymStart

### 5.2.64 Csm_SymKeyWrapAsymUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyWrapAsymUpdate`** `(Csm_ConfigIdType cfgId, uint8 *dataPtr, uint32 *dataLength)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| dataLength | 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 provided buffer. When the request has finished, the actual length of the returned wrapped key shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the symmetrical key wrapping service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-72 Csm_SymKeyWrapAsymUpdate

### 5.2.65 Csm_SymKeyWrapAsymFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_SymKeyWrapAsymFinish`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the symmetrical key wrapping service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-73 Csm_SymKeyWrapAsymFinish

### 5.2.66 Csm_AsymPublicKeyExtractStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPublicKeyExtractStart`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the public key extraction service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-74 Csm_AsymPublicKeyExtractStart

### 5.2.67 Csm_AsymPublicKeyExtractUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPublicKeyExtractUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the public key extraction service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-75 Csm_AsymPublicKeyExtractUpdate

## 5.2.68 Csm_AsymPublicKeyExtractFinish

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPublicKeyExtractFinish`** `(Csm_ConfigIdType cfgId, Csm_AsymPublicKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to a structure where the result (i.e. the asymmetrical public key) is stored in. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the public key extraction service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-76 Csm_AsymPublicKeyExtractFinish

## 5.2.69 Csm_AsymPrivateKeyExtractStart

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyExtractStart`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the private key extraction service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |

| Call Context |
|---|
| > This function can be called from task level only. |

Table 5-77 Csm_AsymPrivateKeyExtractStart

### 5.2.70 Csm_AsymPrivateKeyExtractUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyExtractUpdate`** `(Csm_ConfigIdType cfgId, const uint8 *dataPtr, uint32 dataLength)` | |
| **Parameter** | |
| 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. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the private key extraction service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-78 Csm_AsymPrivateKeyExtractUpdate

### 5.2.71 Csm_AsymPrivateKeyExtractFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyExtractFinish`** `(Csm_ConfigIdType cfgId, Csm_AsymPrivateKeyType *keyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to a structure where the result (i.e. the asymmetrical private key) is stored in. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the private key extraction service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-79 Csm_AsymPrivateKeyExtractFinish

### 5.2.72 Csm_AsymPrivateKeyWrapSymStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyWrapSymStart`** `(Csm_ConfigIdType cfgId, const Csm_AsymPrivateKeyType *keyPtr, const Csm_SymKeyType *wrappingkeyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the private key to be wrapped. |
| wrappingkeyPtr | Holds a pointer to the public key used for wrapping. |
| **Return code** | |
| CSM_E_OK<br>CSM_E_NOT_OK<br>CSM_E_BUSY | Request successful.<br>Request failed.<br>Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the asymmetrical key wrapping service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-80 Csm_AsymPrivateKeyWrapSymStart

### 5.2.73 Csm_AsymPrivateKeyWrapSymUpdate

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyWrapSymUpdate`** `(Csm_ConfigIdType cfgId, uint8 *dataPtr, uint32 *dataLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| 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 provided buffer. When the request has finished, the actual length of the returned wrapped key shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the asymmetrical key wrapping service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-81 Csm_AsymPrivateKeyWrapSymUpdate

### 5.2.74 Csm_AsymPrivateKeyWrapSymFinish

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyWrapSymFinish`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the asymmetrical key wrapping service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-82 Csm_AsymPrivateKeyWrapSymFinish

### 5.2.75 Csm_AsymPrivateKeyWrapAsymStart

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyWrapAsymStart`** `(Csm_ConfigIdType cfgId, const Csm_AsymPrivateKeyType *keyPtr, const Csm_AsymPublicKeyType *wrappingkeyPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| keyPtr | Holds a pointer to the symmetric key to be wrapped. |
| wrappingkeyPtr | Holds a pointer to the key used for wrapping. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to initialize the asymmetrical key wrapping service of the CSM module. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-83 Csm_AsymPrivateKeyWrapAsymStart

### 5.2.76 Csm_AsymPrivateKeyWrapAsymUpdate

| **Prototype** | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyWrapAsymUpdate`** `(Csm_ConfigIdType cfgId, uint8 *dataPtr, uint32 *dataLengthPtr)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| 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. |
| 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 provided buffer. When the request has finished, the actual length of the returned wrapped key shall be stored. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to feed the asymmetrical key wrapping service with the input data. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-84 Csm_AsymPrivateKeyWrapAsymUpdate

### 5.2.77 Csm_AsymPrivateKeyWrapAsymFinish

| Prototype | |
|---|---|
| `Csm_ReturnType` **`Csm_AsymPrivateKeyWrapAsymFinish`** `(Csm_ConfigIdType cfgId)` | |
| **Parameter** | |
| cfgId | Holds the identifier of the CSM module configuration that has to be used during the operation. |
| **Return code** | |
| CSM_E_OK | Request successful. |
| CSM_E_NOT_OK | Request failed. |
| CSM_E_BUSY | Request failed, service is still busy. |
| **Functional Description** | |
| This interface shall be used to finish the asymmetrical key wrapping service. | |
| **Particularities and Limitations** | |
| > This function can be synchronous or asynchronous.<br>> This function is non-reentrant.<br>> This function is called by application. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-85 Csm_AsymPrivateKeyWrapAsymFinish

## 5.3 Services used by CSM

In the following table services provided by other components, which are used by the CSM are listed. For details about prototype and functionality refer to the documentation of the providing component.

| Component | API |
|---|---|
| DET | Det_ReportError |
| CRY | Cry_<Service>Start<br>Cry_<Service>Update<br>Cry_<Service>Finish<br>Cry_<Service>MainFunction<br>Cry_<Service> |

Table 5-86 Services used by the CSM

## 5.4 Callback Functions

This chapter describes the callback functions that are implemented by the CSM and shall be invoked by the CRY modules. The prototypes of the callback functions are provided in the header file `Csm_Cbk.h` by the CSM.

### 5.4.1 Csm_HashCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_HashCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service Hash with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-87 Csm_HashCallbackNotification

### 5.4.2 Csm_HashServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_HashServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service Hash to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-88 Csm_HashServiceFinishNotification

### 5.4.3 Csm_MacGenerateCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_MacGenerateCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service MacGenerate with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-89 Csm_MacGenerateCallbackNotification

### 5.4.4 Csm_MacGenerateServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_MacGenerateServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service MacGenerate to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-90 Csm_MacGenerateServiceFinishNotification

### 5.4.5 Csm_MacVerifyCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_MacVerifyCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service MacVerify with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-91 Csm_MacVerifyCallbackNotification

### 5.4.6 Csm_MacVerifyServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_MacVerifyServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service MacVerify to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-92 Csm_MacVerifyServiceFinishNotification

### 5.4.7 Csm_RandomSeedCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_RandomSeedCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result.<br>`CSM_E_ENTROPY_EXHAUSTION`: request failed, entropy of random number generator is exhausted. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service RandomSeed with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-93 Csm_RandomSeedCallbackNotification

### 5.4.8 Csm_RandomSeedServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_RandomSeedServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service RandomSeed to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-94 Csm_RandomSeedServiceFinishNotification

### 5.4.9 Csm_RandomGenerateCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_RandomGenerateCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result.<br>`CSM_E_ENTROPY_EXHAUSTION`: request failed, entropy of random number generator is exhausted. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service RandomGenerate with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-95 Csm_RandomGenerateCallbackNotification

### 5.4.10 Csm_RandomGenerateServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_RandomGenerateServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service RandomGenerate to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |

| Call Context |
|---|
| > This function can be called from task level only. |

Table 5-96 Csm_RandomGenerateServiceFinishNotification

### 5.4.11 Csm_SymBlockEncryptCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymBlockEncryptCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymBlockEncrypt with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-97 Csm_SymBlockEncryptCallbackNotification

### 5.4.12 Csm_SymBlockEncryptServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymBlockEncryptServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymBlockEncrypt to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-98 Csm_SymBlockEncryptServiceFinishNotification

### 5.4.13 Csm_SymBlockDecryptCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymBlockDecryptCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymBlockDecrypt with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-99 Csm_SymBlockDecryptCallbackNotification

### 5.4.14 Csm_SymBlockDecryptServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymBlockDecryptServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymBlockDecrypt to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-100 Csm_SymBlockDecryptServiceFinishNotification

### 5.4.15 Csm_SymEncryptCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymEncryptCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymEncrypt with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-101 Csm_SymEncryptCallbackNotification

### 5.4.16 Csm_SymEncryptServiceFinishNotification

| **Prototype** | |
|---|---|
| void **Csm_SymEncryptServiceFinishNotification** (void) | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymEncrypt to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-102 Csm_SymEncryptServiceFinishNotification

### 5.4.17 Csm_SymDecryptCallbackNotification

| **Prototype** | |
|---|---|
| void **Csm_SymDecryptCallbackNotification** (Csm_ReturnType Result) | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymDecrypt with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-103 Csm_SymDecryptCallbackNotification

### 5.4.18 Csm_SymDecryptServiceFinishNotification

| Prototype | |
|---|---|
| void **Csm_SymDecryptServiceFinishNotification** (void) | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymDecrypt to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-104 Csm_SymDecryptServiceFinishNotification

### 5.4.19 Csm_AsymEncryptCallbackNotification

| Prototype | |
|---|---|
| void **Csm_AsymEncryptCallbackNotification** (Csm_ReturnType Result) | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service AsymEncrypt with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-105 Csm_AsymEncryptCallbackNotification

### 5.4.20 Csm_AsymEncryptServiceFinishNotification

| Prototype | |
|---|---|
| `void Csm_AsymEncryptServiceFinishNotification (void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service AsymEncrypt to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-106 Csm_AsymEncryptServiceFinishNotification

### 5.4.21 Csm_AsymDecryptCallbackNotification

| Prototype | |
|---|---|
| `void Csm_AsymDecryptCallbackNotification (Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service AsymDecrypt with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-107 Csm_AsymDecryptCallbackNotification

### 5.4.22 Csm_AsymDecryptServiceFinishNotification

| **Prototype** | |
|---|---|
| `void Csm_AsymDecryptServiceFinishNotification (void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service AsymDecrypt to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-108 Csm_AsymDecryptServiceFinishNotification

### 5.4.23 Csm_SignatureGenerateCallbackNotification

| **Prototype** | |
|---|---|
| `void Csm_SignatureGenerateCallbackNotification (Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SignatureGenerate with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |

| Call Context |
|---|
| > This function can be called from task level only. |

Table 5-109 Csm_SignatureGenerateCallbackNotification

### 5.4.24 Csm_SignatureGenerateServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SignatureGenerateServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SignatureGenerate to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-110 Csm_SignatureGenerateServiceFinishNotification

### 5.4.25 Csm_SignatureVerifyCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_SignatureVerifyCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SignatureVerify with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-111 Csm_SignatureVerifyCallbackNotification

### 5.4.26 Csm_SignatureVerifyServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_SignatureVerifyServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SignatureVerify to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-112 Csm_SignatureVerifyServiceFinishNotification

### 5.4.27 Csm_ChecksumCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_ChecksumCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service Checksum with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-113 Csm_ChecksumCallbackNotification

### 5.4.28 Csm_ChecksumServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_ChecksumServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service Checksum to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-114 Csm_ChecksumServiceFinishNotification

### 5.4.29 Csm_KeyDeriveCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_KeyDeriveCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service KeyDerive with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-115 Csm_KeyDeriveCallbackNotification

### 5.4.30 Csm_KeyDeriveServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_KeyDeriveServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service KeyDerive to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-116 Csm_KeyDeriveServiceFinishNotification

### 5.4.31 Csm_KeyDeriveSymKeyCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_KeyDeriveSymKeyCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service KeyDeriveSymKey with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-117 Csm_KeyDeriveSymKeyCallbackNotification

### 5.4.32 Csm_KeyDeriveSymKeyServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_KeyDeriveSymKeyServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service KeyDeriveSymKey to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-118 Csm_KeyDeriveSymKeyServiceFinishNotification

### 5.4.33 Csm_KeyExchangeCalcPubValCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_KeyExchangeCalcPubValCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service KeyExchangeCalcPubVal with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-119 Csm_KeyExchangeCalcPubValCallbackNotification

### 5.4.34 Csm_KeyExchangeCalcPubValServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_KeyExchangeCalcPubValServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service KeyExchangeCalcPubVal to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-120 Csm_KeyExchangeCalcPubValServiceFinishNotification

### 5.4.35 Csm_KeyExchangeCalcSecretCallbackNotification

| Prototype | |
|---|---|
| void **Csm_KeyExchangeCalcSecretCallbackNotification** (Csm_ReturnType Result) | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service KeyExchangeCalcSecret with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-121 Csm_KeyExchangeCalcSecretCallbackNotification

### 5.4.36 Csm_KeyExchangeCalcSecretServiceFinishNotification

| Prototype | |
|---|---|
| void **Csm_KeyExchangeCalcSecretServiceFinishNotification** (void) | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service KeyExchangeCalcSecret to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-122 Csm_KeyExchangeCalcSecretServiceFinishNotification

### 5.4.37 Csm_KeyExchangeCalcSymKeyCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_KeyExchangeCalcSymKeyCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service KeyExchangeCalcSymKey with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-123 Csm_KeyExchangeCalcSymKeyCallbackNotification

### 5.4.38 Csm_KeyExchangeCalcSymKeyServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_KeyExchangeCalcSymKeyServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service KeyExchangeCalcSymKey to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-124 Csm_KeyExchangeCalcSymKeyServiceFinishNotification

### 5.4.39 Csm_SymKeyExtractCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymKeyExtractCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymKeyExtract with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-125 Csm_SymKeyExtractCallbackNotification

### 5.4.40 Csm_SymKeyExtractServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymKeyExtractServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymKeyExtract to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-126 Csm_SymKeyExtractServiceFinishNotification

### 5.4.41 Csm_SymKeyWrapSymCallbackNotification

| Prototype | |
|---|---|
| `void` **`Csm_SymKeyWrapSymCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymKeyWrapSym with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-127 Csm_SymKeyWrapSymCallbackNotification

### 5.4.42 Csm_SymKeyWrapSymServiceFinishNotification

| Prototype | |
|---|---|
| `void` **`Csm_SymKeyWrapSymServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymKeyWrapSym to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-128 Csm_SymKeyWrapSymServiceFinishNotification

### 5.4.43 Csm_SymKeyWrapAsymCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymKeyWrapAsymCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service SymKeyWrapAsym with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-129 Csm_SymKeyWrapAsymCallbackNotification

### 5.4.44 Csm_SymKeyWrapAsymServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_SymKeyWrapAsymServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service SymKeyWrapAsym to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-130 Csm_SymKeyWrapAsymServiceFinishNotification

### 5.4.45 Csm_AsymPublicKeyExtractCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_AsymPublicKeyExtractCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service AsymPublicKeyExtract with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-131 Csm_AsymPublicKeyExtractCallbackNotification

### 5.4.46 Csm_AsymPublicKeyExtractServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_AsymPublicKeyExtractServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service AsymPublicKeyExtract to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-132 Csm_AsymPublicKeyExtractServiceFinishNotification

### 5.4.47 Csm_AsymPrivateKeyExtractCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_AsymPrivateKeyExtractCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service AsymPrivateKeyExtract with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-133 Csm_AsymPrivateKeyExtractCallbackNotification

### 5.4.48 Csm_AsymPrivateKeyExtractServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_AsymPrivateKeyExtractServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service AsymPrivateKeyExtract to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-134 Csm_AsymPrivateKeyExtractServiceFinishNotification

### 5.4.49 Csm_AsymPrivateKeyWrapSymCallbackNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_AsymPrivateKeyWrapSymCallbackNotification`** `(Csm_ReturnType Result)` | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service AsymPrivateKeyWrapSym with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-135 Csm_AsymPrivateKeyWrapSymCallbackNotification

### 5.4.50 Csm_AsymPrivateKeyWrapSymServiceFinishNotification

| **Prototype** | |
|---|---|
| `void` **`Csm_AsymPrivateKeyWrapSymServiceFinishNotification`** `(void)` | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service AsymPrivateKeyWrapSym to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-136 Csm_AsymPrivateKeyWrapSymServiceFinishNotification

### 5.4.51 Csm_AsymPrivateKeyWrapAsymCallbackNotification

| **Prototype** | |
|---|---|
| void **Csm_AsymPrivateKeyWrapAsymCallbackNotification** (Csm_ReturnType Result) | |
| **Parameter** | |
| Result | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result. |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall call the callback function as given in the configuration of the service AsymPrivateKeyWrapAsym with the argument given by Result. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-137 Csm_AsymPrivateKeyWrapAsymCallbackNotification

### 5.4.52 Csm_AsymPrivateKeyWrapAsymServiceFinishNotification

| **Prototype** | |
|---|---|
| void **Csm_AsymPrivateKeyWrapAsymServiceFinishNotification** (void) | |
| **Parameter** | |
| - | |
| **Return code** | |
| - | |
| **Functional Description** | |
| This function shall set the state of the service AsymPrivateKeyWrapAsym to idle. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by cryptographic primitive. | |
| Call Context | |
| > This function can be called from task level only. | |

Table 5-138 Csm_AsymPrivateKeyWrapAsymServiceFinishNotification

### 5.5 Configurable Interfaces

#### 5.5.1 Notifications

At its configurable interfaces the CSM defines notifications that can be mapped to callback functions provided by other modules. This only applies for the asynchronous processing mode. The mapping is not statically defined by the CSM but can be performed at configuration time. For each service, a notification can be configured. The appropriate function prototype signature is described in the following sub-chapters. The name of the function is only a placeholder.

**ServiceCallback**

| Prototype | |
|---|---|
| `Std_ReturnType` **`ServiceCallback`** `(Csm_ReturnType Return)` | |
| **Parameter** | |
| `Return` | Contains the result of a cryptographic operation.<br>`CSM_E_OK`: request successful.<br>`CSM_E_NOT_OK`: request failed.<br>`CSM_E_BUSY`: request failed, service is still busy.<br>`CSM_E_SMALL_BUFFER`: provided buffer is too small to store the result.<br>`CSM_E_ENTROPY_EXHAUSTION`: request failed, entropy of random number generator is exhausted. |
| **Return code** | |
| `E_OK`<br>`E_NOT_OK` | Return Value is ignored in this implementation of the Csm |
| **Functional Description** | |
| Function will be called when configured service has finished. | |
| **Particularities and Limitations** | |
| > This function is synchronous.<br>> This function is non-reentrant.<br>> This function is called by Csm. | |
| Call Context | |
| > This function will be called from task level only. | |

Table 5-139 ServiceCallback

### 5.6 Service Ports

#### 5.6.1 Client Server Interface

A client server interface is related to a Provide Port at the server side and a Require Port at client side.

#### 5.6.2 Provide Ports on CSM Side

At the Provide Ports of the Csm the cryptographic API functions described in 5.2 are available as Runnable Entities. The Runnable Entities are invoked via Operations. The mapping from a SWC client call to an Operation is performed by the RTE. In this mapping the RTE adds Port Defined Argument Values to the client call of the SWC, if configured.

# 6. Configuration

In the Csm the attributes can be configured with the following tools:

> Configuration in DaVinci Configurator

**FAQ**
By default the CSM configuration is empty. To create a service instance, the specific service sub container has to be created. Afterwards you can instance the service by creating a new configuration container.

## 6.1 Configuration Variants

The CSM supports the configuration variants

> `VARIANT-PRE-COMPILE`

## 6.2 Configuration with DaVinci Configurator 5

### 6.2.1 Common Properties

| Attribute Name | Values<br>Default value is typed bold | Description |
|---|---|---|
| CsmDevErrorDetect | **STD_ON**<br>STD_OFF | Pre-processor switch to enable and disable development error detection.<br>True: Development error detection enabled.<br>False: Development error detection disabled |
| CsmDisableNotConfiguredApis | **STD_ON**<br>STD_OFF | If enabled, APIs of not configured services will be disabled. |
| CsmMainFunctionPeriod | 0.001 to 65.535 | Specifies the period of main function Csm_MainFunction in seconds. |
| CsmMaxAlignScalarType | 8<br>16<br>**32** | The scalar type which has the maximum alignment restrictions on the given platform.<br>This type can be e.g. uint8, uint16 or uint32. |
| CsmMaximumBlockingTime | 1 to 4294967295 | If interruption is turned on with the configuration option CsmUseInterruption, this option configures the maximum time in microseconds the main function shall be allowed to run before it must interrupt itself. The lowest allowed value for the option is implementation dependent.<br><br>NOT USED |
| CsmRteBufferSize | 1 to 4294967295 ; **128** | Specifies the size in bytes for the Rte Buffer types created by Csm. |

| Attribute Name | Values<br>Default value is typed bold | Description |
|---|---|---|
| CsmUseInterruption | STD_ON<br>**STD_OFF** | Pre-processor switch to enable and disable interruption of job processing.<br><br>NOT USED<br>True: Interruption of job processing enabled<br>False: Interruption of job processing disabled |
| CsmUseSyncJobProcessing | **STD_ON**<br>STD_OFF | Pre-processor switch to enable and disable synchronous job processing.<br>True: synchronous job processing enabled<br>False: synchronous job processing disabled |
| CsmUserConfigFile | String | User configuration file that shall be part of the Csm configuration.<br>If you want to overwrite or provide own settings in the generated configuration file, you can specify a path to a user defined configuration file. The user defined configuration file will be included at the end of the generated file. Thus definitions in the user defined configuration file can overwrite definitions in the generated configuration file. |
| CsmVersionInfoApi | **STD_ON**<br>STD_OFF | Pre-processor switch to enable and disable availability of the API Csm_GetVersionInfo().<br>True: API Csm_GetVersionInfo() is available.<br>False: API Csm_GetVersionInfo() is not available. |

### 6.2.2 Service Type related Properties

Depending on the type of service, the following parameter may configurable:

| Attribute Name | Values<br>Default value is typed bold | Description |
|---|---|---|
| Csm<ServiceType>MaxKeySize | 1..<br>4294967295 | This is the maximum size over all key lengths used in all CRY primitives, which implement the specific kind of <ServiceType>.<br>Please note that the calling application has to provide the key buffer. So, it has to be ensured that the size of this buffer matches with the configured value here. |

### 6.2.3 Service specific Properties

Each service configuration has the following adjustable parameters:

| Attribute Name | Description |
|---|---|
| Csm<ServiceType>Config | This container holds the configuration of one <ServiceType> service. The container name serves as a symbolic name for the identifier of a service configuration. |

| Attribute Name | Description |
|---|---|
| CsmCallback<ServiceType> | Callback function to be called if service has finished. This parameter is only needed if the CSM is in asynchronous mode. |
| Csm<ServiceType>IncludeFile | Header file of the underlying cryptographic service that shall be used. |
| Csm<ServiceType>InitConfiguration | This is the name of the C symbol, which contains the configuration of the underlying cryptographic primitive.<br>Usually, this symbol represents a structure provided by the CRY module. |
| Csm<ServiceType>PrimitiveName | This is the name of the cryptographic primitive to use.<br>This name will be used to form the function pointers to the Start, Update and Finish functions of the corresponding cryptographic primitive according to the following rule:<br><name>[Start\|Update\|Finish]<br>Usually these functions are provided by the CRY module. |
| Csm<ServiceType>UseServicePorts | This parameter defines if this service is accessible via service ports. The PortName will be derived from the service name. |
| Csm<ServiceType>CryRef | Reference to MICROSAR CRY. This eases up the configuration for MICROSAR CRY. All necessary attributes will be set automatically if linked with a CRY service instance. |

**Usage of callback functions without the RTE**
The default use case of the CSM is the use with the RTE, so the callback functions are automatically set to Rte_Call_<Shortname>_Callback_JobFinished. To use the callback function without the RTE set this field to user defined.

# 7. AUTOSAR Standard Compliance

## 7.1 Deviations

The current implementation does not have any deviations.

## 7.2 Additions/ Extensions

### 7.2.1 Not supported service APIs can be disabled

When enabling the switch “Disable not used APIs”, each API of a service without a configuration will be disabled.

## 7.3 Memory Initialization

Not every start-up code of embedded targets and neither CANoe-Emulation provide initialized RAM. It thus may happen that the state of a variable that needs initialized RAM may not be set to the expected initial value. Therefore an explicit initialization of such variables has to be provided at start-up by calling the additional function Csm_InitMemory.

For more information refer to chapter 3.2 ‘Initialization’.

## 7.4 Limitations

### 7.4.1 Interruption of job processing

The interruption of job processing is not supported in this implementation of the CSM. The API `Csm_Interruption` can be activated for compatibility reasons but has no effect when called.

### 7.4.2 Production Error Reporting

Currently, no production errors are reported.

### 7.4.3 Development Error Reporting

According to SWS [1], the CSM module has six different Error Codes. The current implementation only reports four. CSM_E_PARAM_KEY_TYPE_INVALID and CSM_E_BUFFER_TOO_SMALL are not reported.

# 8. Glossary and Abbreviations

## 8.1 Glossary

| Term | Description |
|---|---|
| Cryptographic Primitive | An underlying cryptographic module or library |

Table 8-1 Glossary

## 8.2 Abbreviations

| Abbreviation | Description |
|---|---|
| API | Application Programming Interface |
| AUTOSAR | Automotive Open System Architecture |
| BSW | Basis Software |
| Csm | Crypto Service Manager |
| DEM | Diagnostic Event Manager |
| DET | Development Error Tracer |
| ECU | Electronic Control Unit |
| HIS | Hersteller Initiative Software |
| MICROSAR | Microcontroller Open System Architecture (the Vector AUTOSAR solution) |
| RTE | Runtime Environment |
| SchM | Schedule Manager |
| SRS | Software Requirement Specification |
| SWC | Software Component |
| SWS | Software Specification |

Table 8-2 Abbreviations

# 9. Contact

Visit our website for more information on

> News
> Products
> Demo software
> Support
> Training data
> Addresses

www.vector.com