Elektrobit

# EB tresos® AutoCore Generic 8 Crypto and Security Stack documentation

product release 8.8.4

Elektrobit Automotive GmbH
Am Wolfsmantel 46
91058 Erlangen, Germany
Phone: +49 9131 7701 0
Fax: +49 9131 7701 6333
Email: info.automotive@elektrobit.com

## Technical support

https://www.elektrobit.com/support

## Legal disclaimer

Confidential information.

# Table of Contents

# 1. Overview of EB tresos AutoCore Generic 8 Crypto and Security Stack documentation

Welcome to the EB tresos AutoCore Generic 8 Crypto and Security Stack (ACG8 Crypto and Security Stack) product documentation.

This document provides:

- Chapter 2, “Supported features“: list of features supported by the ACG8 Crypto and Security Stack
- Chapter 3, “ACG8 Crypto and Security Stack release notes“: release notes for the ACG8 Crypto and Security Stack modules
- Chapter 4, “ACG8 Crypto and Security Stack user guide“: background information and instructions
- Chapter 5, “ACG8 Crypto and Security Stack module references“: information about configuration parameters and the application programming interface

# 2. Supported features

## 2.1. Overview

This chapter provides an overview of the products of ACG8 Crypto and Security Stack and the features that are currently supported.

Section 2.2, “Product details” contains an overview of the products of ACG8 Crypto and Security Stack.

Section 2.3.1, “Supported CryIf features” contains an overview of `CryIf` features.

Section 2.3.2, “Supported Csm features” contains an overview of `Csm` features.

Section 2.3.3, “Supported SecOC features” contains an overview of `SecOC` features.

## 2.2. Product details

ACG8 Crypto and Security Stack provides AUTOSAR modules for the EB tresos AutoCore Generic (ACG) product line. The modules are based on AUTOSAR 4.3.0, selected features of AUTOSAR 4.3.1, and EB-specific enhancements implemented compatible to the AUTOSAR standard.

ACG8 Crypto and Security Stack includes the following basic software modules:

| **Basic software modules** | **Module abbreviation** |
|---|---|
| Crypto Interface | CryIf |
| Crypto Service Manager | Csm |
| Secure Onboard Communication | SecOC |

## 2.3. Feature details

This chapter contains an overview of the supported and unsupported features.

### 2.3.1. Supported CryIf features

ACG8 CRYIF provides the following main features according to the AUTOSAR specification:

- Standardized interface to `Csm` and Crypto Driver modules to manage different crypto hardware and software solutions like HSM, SHE or software-based complex device drivers
- Unique interface to manage multiple Crypto Driver modules with a single `Csm`
- Maintenance of a mapping scheme of the various crypto solutions for use by the `Csm`
- Copy keys from one Crypto Driver to another by using an internal buffer with configurable size

## 2.3.2. Supported Csm features

ACG8 CSM provides the following main features according to the AUTOSAR specification:

- **Provision of synchronous and asynchronous services to enable a unique access to basic cryptographic functionalities**
- **Standardized interfaces to the following cryptographic functions:**
  - Hash code generation
  - Message authentication code (MAC) generation and verification
  - Random number generation
  - Authenticated encryption with associated data
  - Signature generation and verification
  - Key management
  - Cipher services
- **Job handling**
  - Priority-based job queuing
  - Cancellation of ongoing job requests
- **Possibility to include different cryptographic algorithms via Crypto Driver module:**
  - According to the AUTOSAR Crypto Service Manager specification, the actual cryptographic algorithms are contained in a separate Crypto Driver module, which is included and accessed by the Crypto Service Manager via the Crypto Interface module.

## 2.3.3. Supported SecOC features

ACG8 SECOC provides the following main features according to the AUTOSAR specification:

- **Direct interface, transport protocol, and triggered transmission:** ACG8 SECOC can be configured to interact with a direct communication interface, a transport protocol or a triggered transmission on the

ECU bus using e.g. CAN or FlexRay. The applications send and receive the data via e.g. the `Com` or the `Dcm` module.

- **Secured PDU collection:** ACG8 SECOC can be configured to send the secured PDU as standard secured PDU or as a PDU collection. If a secured PDU is configured for secured PDU collection, the secured PDU is sent or received within two separate PDUs: an authenticated PDU containing the authentic data and a cryptographic PDU containing the authentication information and an optional message linker.
- **External freshness source:** ACG8 SECOC queries the freshness values required for generation or verification of secured PDUs from an external freshness source, e.g. a freshness management SWC. ACG8 SECOC can be configured to request the freshness values either via an Rte port if the request is directed at a software component or via a C function if the request is directed at a complex driver.
- **Synchronous and asynchronous crypto functionality:** ACG8 SECOC can be configured per PDU to use the product ACG8 CSM synchronously or asynchronously for cryptographic operations, e.g. MAC generation or verification.
- **Application indication:** ACG8 SECOC verifies received PDU messages and if it detects any fault, the PDU is rejected. This happens completely transparent to the receiver. To inform the receiver about such a verification error, a callback function can be registered to get this verification error indicated on application side.

  This feature can be extended with a callback function that indicates a failure in the MAC generation process to the application.
- **Support for overriding the verification status:** ACG8 SECOC provides an interface to override the verification status when receiving a secured PDU. It can be overridden either with *fail* or *pass*. Depending on the verification status, the secured PDU is either dropped or passed to the upper layer.
- **Default MAC:** ACG8 SECOC provides a configuration parameter to send out secured PDUs with a default MAC in case the MAC generation failed on sender side.
- **Support for skipping the PDU verification:** ACG8 SECOC can be configured to either perform or skip the verification of a secured PDU.
- **Secured area:** ACG8 SECOC can be configured to either secure all data of an authentic PDU or a secured area within the authentic PDU. The secured area is defined by an offset and a length. Only the data within the secured area is subject to the cryptographic calculations for a secured PDU.
- **Uniqueness of SecOCDataIds and SecOCFreshnessValueIds is optional:** ACG8 SECOC allows the configuration of SecOCDataIds and SecOCFreshnessValueIds with values that are not unique for each PDU.
- **Support for TxConfirmation time-out:** ACG8 SECOC allows the configuration of the TxConfirmation time-out for every PDU.
- **Support for updating the secured PDU layout:** ACG8 SECOC provides support to configure callout functions that can be used to modify the layout of the secured PDU.
- **Support for post-build:** ACG8 SECOC supports post-build loadable and selectable configuration.

# 3. ACG8 Crypto and Security Stack release notes

## 3.1. Overview

This chapter provides the ACG8 Crypto and Security Stack product specific release notes. General release notes that are applicable to all products are provided in the EB tresos AutoCore Generic documentation. Refer to the general release notes in addition to the product release notes documented here.

## 3.2. Scope of the release

### 3.2.1. Configuration tool

Your release of EB tresos AutoCore is compatible with the release of the EB tresos Studio configuration tool:

- EB tresos Studio: 28.2.0 b211016-0103

### 3.2.2. AUTOSAR modules

The following table lists the AUTOSAR modules that are part of this ACG8 Crypto and Security Stack release.

| Module name | AUTOSAR version and revision | SWS version and revision | Module version | Supplier |
|---|---|---|---|---|
| CryIf | 4.3.0 [] | 4.3.0 [0000] | 1.0.27 | Elektrobit Automotive GmbH |
| Csm | 4.3.0 [] | 4.3.0 [0000] | 3.1.15 | Elektrobit Automotive GmbH |
| SecOC | 4.3.0 [] | 4.3.0 [0000] | 2.7.6 | Elektrobit Automotive GmbH |

Table 3.1. Hardware-Independent Modules specified by the AUTOSAR standard

### 3.2.3. EB (Elektrobit) modules

The following table lists all modules which are part of this release but are not specified by the AUTOSAR standard. These modules include tooling developed by EB or they may hold files shared by all other modules.

| Module name | Module version | Supplier |
|---|---|---|
| No EB modules available | | |

Table 3.2. Modules not specified by the AUTOSAR standard

### 3.2.4. MCAL modules and EB tresos AutoCore OS

For information about MCAL modules and OS, refer to the respective documentation, which is available as PDF at `$TRESOS_BASE/doc/3.0_EB_tresos_AutoCore_OS` and `$TRESOS_BASE/doc/5.0_MCAL_modules`[1]. It is also available in the online help in EB tresos Studio. Browse to the folders `EB tresos AutoCore OS` and `MCAL modules`.

## 3.3. Module release notes

### 3.3.1. CryIf module release notes

- AUTOSAR R4.3 Rev 0
- AUTOSAR SWS document version: 4.3.0
- Module version: 1.0.27.B466224
- Supplier: Elektrobit Automotive GmbH

#### 3.3.1.1. Change log

This chapter lists the changes between different versions.

[1] `$TRESOS_BASE` is the location at which you installed EB tresos Studio.

**Module version 1.0.27**
2021-10-08

- Removed the dependency to the not mandatory CommonPublishedInformation.

**Module version 1.0.26**
2021-09-17

- Fixed incorrect query of VendorApiInfix and VendorId.

**Module version 1.0.25**
2021-08-20

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.24**
2021-06-25

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.23**
2021-04-30

- ASCCRYIF-169 Fixed known issue: CryIf causes unexpected data inconsistencies if CryIf_KeyElement-Copy is used for keys which are located in different Crypto drivers.
- Added support for EB tresos HandleIdWizards.

**Module version 1.0.22**
2021-01-22

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.21**
2020-12-18

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.20**
2020-10-23

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.19**
2020-09-25

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.18**
2020-07-31

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.17**
2020-02-21

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.16**
2020-01-24

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.15**
2019-12-06

- Added configuration parameter to switch between CryIf 4.3.0 and 4.3.1 API and ARXML compatibility and improved API and ARXML compatibility in general. Also this configuration parameter provides the possibility to choose the mixed 4.3.0 and 4.3.1 EB style API and ARXML version that is necessary for old EB Csm modules less than version 3.1.0 and EB Crypto modules less than version 2.0.0.

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.14**
2019-10-11

- Internal module improvement. This module version update does not affect module functionality.

**Module version 1.0.13**
2019-08-09

- ASCCRYIF-103 Fixed known issue: CryIf does not generate symbolic names for CryIfChannels and CryIfKeys
- ASCCRYIF-104 Fixed known issue: CryIf does not use symbolic names for referenced CryptoDriverObjects and CryptoKeys

**Module version 1.0.12**
2019-06-19

- Added creation of Crypto API Module implementation prefix based on BSWMDs in addition to the default creation based on CommonPublishedInformations.

**Module version 1.0.11**
2019-05-17

- Removed 'myEcuParameterDefinition' from XDM and BMD file.
- ASCCRYIF-101 Fixed known issue: DESTINATION-REFs in the VSMD violate TPS_ECUC_06015

**Module version 1.0.10**
2019-01-25

- Changed return values of CryIf_KeyElementCopy() and CryIf_KeyCopy() to CRYPTO_E_KEY_SIZE_MISMATCH instead of E_NOT_OK when the key element sizes do not match, as discussed in https://bugzilla.autosar.org/show_bug.cgi?id=79493 and realized in R4.4.

**Module version 1.0.9**
2018-10-26

- Internal module improvement. This module version update does not affect module functionality

**Module version 1.0.8**
2018-06-22

- Improved robustness of CryIf_ProcessJob() and CryIf_CancelJob() regarding invalid key IDs

**Module version 1.0.7**
2018-05-25

- Internal module improvement. This module version update does not affect module functionality

**Module version 1.0.6**
2018-04-06

- ASCCRYIF-67 Fixed known issue: The KeyCopy / KeyElementCopy functions fail to copy key elements
- ASCCRYIF-71 Fixed known issue: Incorrect check of referenced functions in KeyDerive, KeyCopy, KeyElementCopy and CertificateVerify

**Module version 1.0.5**
2018-03-16

- Internal module improvement. This module version update does not affect module functionality

**Module version 1.0.4**
2018-02-16

- Internal module improvement. This module version update does not affect module functionality

**Module version 1.0.3**
2017-12-20

- Corrected compiler warnings
- Improved robustness of multi-instantiation of Crypto Drivers regarding Crypto preconfiguration and relative x-paths

**Module version 1.0.2**
2017-11-17

- Updated limitations and documentation

**Module version 1.0.1**
2017-10-02

- ASCCRYIF-18 Fixed known issue: Number of configurable keys and channels is limited to 32
- ASCCRYIF-16 Fixed known issue: CryIf_ProcessJob() and CryIf_CancelJob() pass CryIf channel ID instead of Crypto driver object ID to Crypto API
- ASCCRYIF-15 Fixed known issue: CryIf routes Csm API calls to wrong Crypto modules and/or CryptoDriverObjects and/or CryptoKeys

**Module version 1.0.0**
2017-08-04

- Implemented CryIf module compliant to the AUTOSAR 4.3 specification

#### 3.3.1.2. New features

- No new features have been added since the last release.

#### 3.3.1.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- This module provides no EB-specific enhancements.

#### 3.3.1.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- CryIfKeyId does not start from zero

  Description:

CryIfKeyId shall be consecutive, gapless and shall start from zero.

Rationale:

- This requirement is not applicable. It's invalidated by note 'The Ids in the configuration containers shall be consecutive, gapless and shall start from zero'. It's replaced by requirement ECUC_CryIf_00007_CORRECTION.

Requirements:

ECUC_CryIf_00007

- CryIfChannelId does not start from zero

Description:

CryIfChannelId shall be consecutive, gapless and shall start from zero.

Rationale:

- This requirement is not applicable. It's invalidated by note 'The Ids in the configuration containers shall be consecutive, gapless and shall start from zero'. It's replaced by requirement ECUC_CryIf_00004_CORRECTION.

Requirements:

ECUC_CryIf_00004

- Return value of CryIf_KeyCopy() and CryIf_KeyElementCopy()

Description:

The functions CryIf_KeyCopy() and CryIf_KeyElementCopy() now return CRYPTO_E_KEY_SIZE_MISMATCH instead of E_NOT_OK when the key element sizes do not match.

Rationale:

- https://bugzilla.autosar.org/show_bug.cgi?id=79493

Requirements:

SWS_CryIf_00115, SWS_CryIf_00121

#### 3.3.1.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Job Cancellation Interface: CryIf_CancelJob() expects Crypto Drivers with the following Crypto_CancelJob API: Std_ReturnType Crypto_CancelJob( uint32 objectId, Crypto_JobType* job ). Also see RfC 80287.

#### 3.3.1.6. Open-source software

`CryIf` does not use open-source software.

## 3.3.2. Csm module release notes

- AUTOSAR R4.3 Rev 0
- AUTOSAR SWS document version: 4.3.0
- Module version: 3.1.15.B466224
- Supplier: Elektrobit Automotive GmbH

#### 3.3.2.1. Change log

This chapter lists the changes between different versions.

**Module version 3.1.15**
2021-09-17

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.1.13**
2021-06-25

- Add configuration check to ensure that the queue and key referenced in a Csm Job are refering to the same Crypto driver.

**Module version 3.1.12**
2021-05-28

- Added justifications for tasking compiler warnings and fixed a compiler warning.

**Module version 3.1.11**
2021-04-30

- ASCCSM-473 Fixed known issue: Placing the Csm plugin in another directory than <tresos>/plugins is not possible.
- Added support for EB tresos HandleIdWizards.

**Module version 3.1.8**
2021-01-22

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.1.7**
2020-12-18

- Adjusted Code-Metric Deviation rule texts to follow specified syntax.
- Fixed availability of the declartion for Csm_CancelJob, if all jobs with enabled RTE usage only reference primitives of service CRYPTO_RANDOMGENERATE.

**Module version 3.1.6**
2020-10-23

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.1.5**
2020-09-25

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.1.4**
2020-06-19

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.1.3**
2020-05-22

- Added configuration parameter to switch the the implementation of the Client-Server-Operation KeyElementGet of the Client-Server-Interface CsmKeyManagement_{Config} [SWS_Csm_01905] to be compliant with the original AUTOSAR specification or to be correct respective to the specification of Csm_KeyElementGet [SWS_Csm_00959].

**Module version 3.1.2**
2020-03-25

- ASCCSM-407 Fixed known issue: Incorrect queuing of Csm jobs causes negative response or execution on the wrong Crypto Driver Object.

**Module version 3.1.1**
2020-01-24

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.1.0**
2019-12-06

- Added configuration parameter to switch between Csm 4.3.0 and 4.3.1 API and ARXML compatibility and improved API and ARXML compatibility in general. Also this configuration parameter provides the possibility to choose the mixed 4.3.0 and 4.3.1 EB style API and ARXML version that is necessary for old EB Crypto modules less than version 2.0.0.

**Module version 3.0.16**
2019-10-11

- Internal module improvement. This module version update does not affect module functionality.

**Module version 3.0.15**
2019-08-09

- ASCCSM-368 Fixed known issue: Csm does not use symbolic names for referenced CryIfChannels and CryIfKeys.

**Module version 3.0.14**
2019-06-19

- Added open source statement to the release documentation.

**Module version 3.0.13**
2019-05-17

- ASCCSM-363 Fixed known issue: DESTINATION-REFs in the VSMD violate TPS_ECUC_06015.
- Added macro CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE (cRYpto_-...) for identification of key exchange shared value key elements in parallel to the existing misspelled but specified macro CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE (cYRpto_-...).

**Module version 3.0.12**
2019-01-25

- ASCCSM-349 Fixed known issue: Incorrect definition of POSSIBLE-ERROR-REFS for client-server operations SignatureVerify and KeyDerive causes RTE generation errors.

**Module version 3.0.11**
2018-10-30

- Added take over of primitive configuration parameter 'CsmMacVerifyCompareLength' or 'CsmSignatureVerifyCompareLength' in member jobPrimitiveInfo->primitiveInfo->resultLength of the Crypto_JobType data structure of a job, to which a primitive of service 'MacVerify' or 'SignatureVerify' is assigned to.
- ASCCSM-341 Fixed known issue: Csm_CertificateVerify() uses wrong verification CryIf key id.
- Removed unnecessary and wrong CompuMethod 'CM_Csm_ConfigIdType' as well as the reference to this CompuMethod in ImplementationDataType 'Csm_ConfigIdType'.

**Module version 3.0.10**
2018-06-22

- ASCCSM-311 Fixed known issue: CsmCallbacks are only triggered if result is E_OK.

**Module version 3.0.9**
2018-05-25

- ASCCSM-295 Fixed known issue: Crypto primitive SIPHASH cannot be used for Csm service MacGenerate.
- ASCCSM-296 Fixed known issue: RTE ports of CsmCallbacks are generated improperly.

**Module version 3.0.8**
2018-04-20

- Changed the sizes of Implementation Data Types 'Csm_KeyDataType_{Crypto}', 'Csm_SeedDataType_{Crypto}' and 'Csm_PublicValueDataType_{Crypto}' from 'sum' to 'max' of all relevant key element sizes as it is discussed in https://bugzilla.autosar.org/show_bug.cgi?id=78552.

**Module version 3.0.7**
2018-03-16

- ASCCSM-253 Fixed known issue: Variant tags mismatch between Csm and AUTOSAR ECU configuration schema files.

**Module version 3.0.6**
2018-02-16

- ASCCSM-242 Fixed known issue: Csm does not generate correct values for the symbolic names identifiers of the CsmKeyId parameter.
- ASCCSM-255 Fixed known issue: Csm interface generator creates zero-size arrays.

**Module version 3.0.5**
2018-01-19

- ASCCSM-233 Fixed known issue: Compiler warning due to misplaced preprocessor instruction in function Csm_CancelJob.
- ASCCSM-234 Fixed known issue: Out-of-bounds access in function Csm_CancelJob() if no callback is referenced.

**Module version 3.0.4**
2017-12-15

- ASCCSM-207 Fixed known issue: Csm compiler errors occur due to unconditional inclusion of DET header file.
- ASCCSM-223 Fixed known issue: Queue slot not released after dequeuing via Csm_Mainfunction() causes NULL POINTER exception.

**Module version 3.0.3**
2017-11-17

- ASCCSM-195 Fixed known issue: Csm does not generate correct symbolic names for CsmJobId parameters.
- ASCCSM-201 Fixed known issue: Client/server interfaces for CsmPrimitives are generated without existing and referenced implementation data types.

**Module version 3.0.2**
2017-10-02

- ASCCSM-174 Fixed known issue: Definition of internal constant is placed in the wrong memory section.
- ASCCSM-180 Fixed known issue: Csm primitives KeyLength configuration parameters are not considered in all name variations.

**Module version 3.0.1**
2017-09-04

- Changed multiplicity of containers CsmCallbacks and CsmKeys to "1", of container CsmPrimitives to "1..inf" and of parameters CsmAEADDecryptAssociatedDataMaxLength, CsmAEADDecryptCiphertextMaxLength, CsmAEADDecryptPlaintextMaxLength, CsmAEADEncryptAssociatedDataMaxLength, CsmAEADEncryptCiphertextMaxLength, CsmAEADEncryptPlaintextMaxLength, CsmDecryptDataMaxLength, CsmDecryptResultMaxLength, CsmEncryptDataMaxLength, CsmEncryptResultMaxLength, CsmHashDataMaxLength, CsmMacGenerateDataMaxLength, CsmMacVerifyDataMaxLength, CsmSignatureGenerateDataMaxLength and CsmSignatureVerifyDataMaxLength to "1".
- Added Csm_Cbk.h.
- Fixed order of entries in Csm_JobConfigurations global configuration data structure.
- API functions can now be invoked concurrently via RTE.

**Module version 3.0.0**
2017-07-28

- Initial release as AUTOSAR 4.3.0 module

#### 3.3.2.2. New features

- The Csm module provides all non-depracated service APIs and functionality according to AUTOSAR 4.3.

#### 3.3.2.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- Added not specified but necessary configuration parameter

  Description:

  The configuration parameters

  - CsmMacVerify/CsmMacVerifyConfig/CsmMacVerifyAlgorithmKeyLength
  - CsmMacVerify/CsmMacVerifyConfig/CsmMacVerifyAlgorithmMode
  - CsmMacVerify/CsmMacVerifyConfig/CsmMacVerifyAlgorithmModeCustom
  - CsmEncrypt/CsmEncryptConfig/CsmEncryptAlgorithmKeyLength
  - CsmSignatureVerify/CsmSignatureVerifyConfig/CsmSignatureVerifyKeyLength

  are added to complete the set of necessary configuration options. See Autosar Bugzilla entries

  - https://www.autosar.org/bugzilla/show_bug.cgi?id=77271
  - https://www.autosar.org/bugzilla/show_bug.cgi?id=78276
  - https://www.autosar.org/bugzilla/show_bug.cgi?id=78327

  Rationale:

  The SWS specifies an incomplete set of Csm configuration parameters.

- Added additional DET checks

  Description:

  The following DET checks

  - jobID is out of range [=> CSM_E_PARAM_HANDLE]
  - configured service of job references by jobID did not match services designated by API function [=> CSM_E_SERVICE_NOT_IDENTICAL (0xE1)]

  are added to enhance the set of meaningful DET checks.

  Rationale:

The SWS specifies an potentially incomplete set of Csm DET checks.

#### 3.3.2.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- Size of Csm_KeyDataType_{Crypto}, Csm_SeedDataType_{Crypto} and Csm_PublicValue-DataType_{Crypto}

  Description:

  The sizes of Implementation Data Types 'Csm_KeyDataType_{Crypto}', 'Csm_SeedDataType_{Crypto}' and 'Csm_PublicValueDataType_{Crypto}' is changed from 'sum' to 'max' of all relevant key element sizes.

  Rationale:

  - https://bugzilla.autosar.org/show_bug.cgi?id=78552

  Requirements:

  SWS_Csm_00827, SWS_Csm_00828, SWS_Csm_00829
- Usage of DEM

  Description:

  The Dem module is not used.

  Rationale:

  - https://bugzilla.autosar.org/show_bug.cgi?id=80231

  Requirements:

  SWS_Csm_00486
- Variation of 'Primitive' and 'Crypto'

  Description:

  The variations for '{Primitive}' and '{Crypto}' of 'Client-Server-Interfaces', 'Implementation Data Types' and 'Ports' were corrected regarding specification errors.

  Rationale:

  - https://bugzilla.autosar.org/show_bug.cgi?id=77966, point 12) of problem description

  Requirements:

SWS_Csm_00946, SWS_Csm_009000, SWS_Csm_00936, SWS_Csm_00947, SWS_Csm_01906, SWS_Csm_01910, SWS_Csm_01915, SWS_Csm_00903, SWS_Csm_00943, SWS_Csm_00902, SWS_Csm_01920, SWS_Csm_00912, SWS_Csm_00935, SWS_Csm_00927, SWS_Csm_00802, SWS_Csm_00803, SWS_Csm_01921, SWS_Csm_01922, SWS_Csm_01923, SWS_Csm_01924, SWS_Csm_01925, SWS_Csm_01928, SWS_Csm_01927, SWS_Csm_01926, SWS_Csm_00922, SWS_Csm_00923, SWS_Csm_01074, SWS_Csm_01075, SWS_Csm_01083, SWS_Csm_01083___D0002 (second occurence of duplicated SWS_Csm_01083 == SWS_Csm_01077), SWS_Csm_01078, SWS_Csm_01079, SWS_Csm_00930, SWS_Csm_00931, SWS_Csm_00932, SWS_Csm_00934___D0002 (first occurence of duplicated SWS_Csm_00934), SWS_Csm_00933, SWS_Csm_00825, SWS_Csm_00832, SWS_Csm_00833, SWS_Csm_00834, SWS_Csm_00835, SWS_Csm_00838

- Variation of 'Job'

  Description:

  The variation for '{Job}' of 'Ports' was corrected regarding specification errors.

  Rationale:

  - https://bugzilla.autosar.org/show_bug.cgi?id=77966, point 11) of problem description

  Requirements:

  SWS_Csm_00931, SWS_Csm_00932, SWS_Csm_00934___D0002 (first occurence of duplicated SWS_Csm_00934), SWS_Csm_00933, SWS_Csm_00825, SWS_Csm_00832, SWS_Csm_00833, SWS_Csm_00834, SWS_Csm_00835, SWS_Csm_00838

- Corrections

  Description:

  The Csm SWS requirements listed below were corrected regarding individual specification errors.

  Rationale:

  - https://bugzilla.autosar.org/show_bug.cgi?id=76745
  - https://bugzilla.autosar.org/show_bug.cgi?id=76783
  - https://bugzilla.autosar.org/show_bug.cgi?id=76940
  - https://bugzilla.autosar.org/show_bug.cgi?id=76982
  - https://bugzilla.autosar.org/show_bug.cgi?id=76985
  - https://bugzilla.autosar.org/show_bug.cgi?id=77049
  - https://bugzilla.autosar.org/show_bug.cgi?id=77110
  - https://bugzilla.autosar.org/show_bug.cgi?id=77261
  - https://bugzilla.autosar.org/show_bug.cgi?id=77264

- https://bugzilla.autosar.org/show_bug.cgi?id=77267
- https://bugzilla.autosar.org/show_bug.cgi?id=77356
- https://bugzilla.autosar.org/show_bug.cgi?id=77536
- https://bugzilla.autosar.org/show_bug.cgi?id=77710
- https://bugzilla.autosar.org/show_bug.cgi?id=77712
- https://bugzilla.autosar.org/show_bug.cgi?id=77722
- https://bugzilla.autosar.org/show_bug.cgi?id=77723
- https://bugzilla.autosar.org/show_bug.cgi?id=77724
- https://bugzilla.autosar.org/show_bug.cgi?id=77781
- https://bugzilla.autosar.org/show_bug.cgi?id=80071
- https://bugzilla.autosar.org/show_bug.cgi?id=80091

Requirements:

SWS_Csm_00168, SWS_Csm_00803, SWS_Csm_00903, SWS_Csm_00928, SWS_Csm_00934, SWS_Csm_00936, SWS_Csm_00943, SWS_Csm_00947, SWS_Csm_00966, SWS_Csm_00970, SWS_Csm_00992, SWS_Csm_00996, SWS_Csm_01001, SWS_Csm_01008, SWS_Csm_01009, SWS_Csm_01012, SWS_Csm_01013, SWS_Csm_01023, SWS_Csm_01025, SWS_Csm_01026, SWS_Csm_01027, SWS_Csm_01031, SWS_Csm_01035, SWS_Csm_01044, SWS_Csm_01053, SWS_Csm_01074, SWS_Csm_01080, SWS_Csm_01543, SWS_Csm_01905, SWS_Csm_01926, SWS_Csm_01927, SWS_Csm_009000, ECUC_Csm_00015, ECUC_Csm_00051, ECUC_Csm_00076, ECUC_Csm_00084, ECUC_Csm_00111, ECUC_Csm_00119, ECUC_Csm_00172, ECUC_Csm_00183, ECUC_Csm_00188

- Duplicated Requirement Ids

Description:

Duplicated requirement Ids are replaced with new, unique Ids.

Rationale:

- https://bugzilla.autosar.org/show_bug.cgi?id=76440
- https://bugzilla.autosar.org/show_bug.cgi?id=77182

Requirements:

SWS_Csm_00037___D0002, SWS_Csm_00828___D0002, SWS_Csm_00877___D0002, SWS_Csm_-00930___D0002, SWS_Csm_00932___D0002, SWS_Csm_00934___D0002, SWS_Csm_01083___-D0002

- Direction of publicValueLengthPtr changed to INOUT

Description:

Direction for parameter publicValueLengthPtr for Client Server operation KeyExchangeCalcPubVal present in CsmKeyManagement C/S interface is INOUT.

Rationale:

- There exists an inconsistency between SWS_Csm_01905 and SWS_Csm_00966 in terms of the direction for the parameter publicValueLengthPtr for KeyExchangeCalcPubVal operation present in CsmKeyManagement C/S interface.

Requirements:

SWS_Csm_01905

- No implementation of feature 'SecureCounter'

Description:

The Csm feature 'SecureCounter' as specified by the Csm SWS is not implemented. This includes APIs, service interfaces and configurations.

Rationale:

- https://bugzilla.autosar.org/show_bug.cgi?id=77262

Requirements:

SWS_Csm_00998, SWS_Csm_00973, SWS_Csm_00999, SWS_Csm_01000, SWS_Csm_09260, SWS_Csm_00837, SWS_Csm_01009, ECUC_Csm_00030, ECUC_Csm_00101, ECUC_Csm_00102

- No implementation of requirements marked as 'deprecated'

Description:

Csm requirements marked as 'deprecated' as specified by the Csm SWS are not implemented. This inlcudes files, data types, APIs, service interfaces and configurations.

Rationale:

- This is planned for a later release.

Requirements:

SWS_Csm_00006, SWS_Csm_00937, SWS_Csm_00938, SWS_Csm_00939, SWS_Csm_00089, SWS_Csm_00094, SWS_Csm_00101, SWS_Csm_00335, SWS_Csm_00341, SWS_Csm_00348, SWS_Csm_00108, SWS_Csm_00114, SWS_Csm_00121, SWS_Csm_00128, SWS_Csm_00134, SWS_Csm_00141, SWS_Csm_00173, SWS_Csm_00180, SWS_Csm_00187, SWS_Csm_00192, SWS_Csm_00700, SWS_Csm_00199, SWS_Csm_00206, SWS_Csm_00212, SWS_Csm_00665,

SWS_Csm_00221, SWS_Csm_00666, SWS_Csm_00228, SWS_Csm_00234, SWS_Csm_00667, SWS_Csm_00243, SWS_Csm_00668, SWS_Csm_00250, SWS_Csm_00256, SWS_Csm_00669, SWS_Csm_00265, SWS_Csm_00670, SWS_Csm_00272, SWS_Csm_00278, SWS_Csm_00671, SWS_Csm_00287, SWS_Csm_00672, SWS_Csm_00294, SWS_Csm_00300, SWS_Csm_00307, SWS_Csm_00673, SWS_Csm_00314, SWS_Csm_00320, SWS_Csm_00327, SWS_Csm_00436, SWS_Csm_00443, SWS_Csm_00450, SWS_Csm_00418, SWS_Csm_00425, SWS_Csm_00432, SWS_Csm_00149, SWS_Csm_00156, SWS_Csm_00163, SWS_Csm_00455, SWS_Csm_00457, SWS_Csm_00775, SWS_Csm_00776, SWS_Csm_00777, SWS_Csm_00780, SWS_Csm_00781, SWS_Csm_00782, SWS_Csm_00783, SWS_Csm_00784, SWS_Csm_00785, SWS_Csm_00786, SWS_Csm_00787, SWS_Csm_00075, SWS_Csm_00856, SWS_Csm_00857, SWS_Csm_00864, SWS_Csm_00865, SWS_Csm_00867, SWS_Csm_00866, SWS_Csm_00877, SWS_Csm_00875, SWS_Csm_00876, SWS_Csm_00881, SWS_Csm_00882, SWS_Csm_00883, SWS_Csm_00878, SWS_Csm_00879, SWS_Csm_00880, SWS_Csm_00842, SWS_Csm_00843, SWS_Csm_00840, SWS_Csm_00841, SWS_Csm_00871, SWS_Csm_00872, SWS_Csm_00874, SWS_Csm_00873, SWS_Csm_00821, SWS_Csm_00906, SWS_Csm_00907, SWS_Csm_00914, SWS_Csm_00913, SWS_Csm_00916, SWS_Csm_00915, SWS_Csm_00889, SWS_Csm_00888, SWS_Csm_00910, SWS_Csm_00911, CSM.Req.Correction.SWS_Csm_00168, SWS_Csm_91002

- Simplified 'Multiplicities'

  Description:

  The muliplicities specified by the Csm SWS for the containers Csm/CsmCallbacks and Csm/CsmKeys as well as for the parameters CsmAEADDecryptAssociatedDataMaxLength, CsmAEADDecryptCiphertextMaxLength, CsmAEADDecryptPlaintextMaxLength, CsmAEADEncryptAssociatedDataMaxLength, CsmAEADEncryptCiphertextMaxLength, CsmAEADEncryptPlaintextMaxLength, CsmDecryptDataMaxLength, CsmDecryptResultMaxLength, CsmEncryptDataMaxLength, CsmEncryptResultMaxLength, CsmHashDataMaxLength, CsmMacGenerateDataMaxLength, CsmMacVerifyDataMaxLength, CsmSignatureGenerateDataMaxLength and CsmSignatureVerifyDataMaxLength is customized to '1'. The multiplicity of container Csm/CsmPrimitives is changed to '1..*'.

  Rationale:

  - All these configuration objects are necessary to create meaningful and accurate ECU configurations.

  Requirements:

  ECUC_Csm_00818, ECUC_Csm_00040, ECUC_Csm_00056, ECUC_Csm_00137, ECUC_Csm_00146, ECUC_Csm_00147, ECUC_Csm_00154, ECUC_Csm_00155, ECUC_Csm_00158, ECUC_Csm_00159, ECUC_Csm_00160, ECUC_Csm_00162, ECUC_Csm_00163, ECUC_Csm_00165, ECUC_Csm_00169, ECUC_Csm_00175

- CsmDevErrorDetect

  Description:

The 'Default value' of configuration parameter 'CsmDevErrorDetect' is changed to 'true'.

Rationale:

- 'CsmDevErrorDetect' is enabled by default to ease integration.

Requirements:

ECUC_Csm_00001

#### 3.3.2.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- No limitations are reported.

#### 3.3.2.6. Open-source software

`Csm` does not use open-source software.

### 3.3.3. SecOC module release notes

- AUTOSAR R4.3 Rev 0
- AUTOSAR SWS document version: 4.3.0
- Module version: 2.7.6.B466224
- Supplier: Elektrobit Automotive GmbH

#### 3.3.3.1. Change log

This chapter lists the changes between different versions.

**Module version 2.7.6**
2021-10-08

- Add support for the usecase: SecOC_StartOfReception is called with TpSduLength = 0

**Module version 2.7.5**
2021-08-20

- ASCSECOC-579 Fixed known issue: Compile error occurs if only Tx or Rx are configured and EB make files are not used

**Module version 2.7.4**
2021-06-25

- ASCSECOC-562 Fixed known issue: The SecOC calls APIs of other modules in an interrupt context and/or exclusive area

**Module version 2.7.3**
2021-03-05

- Internal module improvement. This module version update does not affect module functionality.

**Module version 2.7.2**
2021-02-12

- ASCSECOC-512 Fixed known issue: Authentic PDU is passed to the upper layer with wrong values

**Module version 2.7.1**
2021-01-22

- Internal module improvement. This module version update does not affect module functionality.

**Module version 2.7.0**
2020-12-18

- Implemented callout function which provides the ability to change the Csm job ID during the run time
- Improved Rx state machine over the retry sequences in case of negative situations
- Improved the global data allocation in the module, moved Rx/Tx global data into the respective SecOC_RxData/SecOC_TxData structures

**Module version 2.6.5**
2020-10-23

- Removed issue generated duo to the missing undef for TS_RELOCATABLE_CFG_ENABLE
- ASCSECOC-414 Fixed known issue: Upper layer authentic Tx PDU is not accepted until SecOC is done processing the current PDU with the same ID

**Module version 2.6.4**
2020-06-19

- ASCSECOC-371 Fixed known issue: The cryptographic Tx PDU can contain a wrong message link
- ASCSECOC-380 Fixed known issue: The cryptographic Tx PDU can contain an incomplete message link
- Implemented option for auto-mapping of the main functions
- Changed NO_INIT memory sections to CLEARED
- Improved the Tx side state machine handling

**Module version 2.6.3**
2020-05-22

- Improved the xdm file by moving EB custom configuration parameter from the "General" tab to "EB General" tab
- ASCSECOC-374 Fixed known issue: SecOC can send unintended messages if the bypass mechanism is activated

**Module version 2.6.2**
2020-04-24

- Updated file name from SecOC_PBCfg.c to SecOC_PBcfg.c.

**Module version 2.6.1**
2020-03-27

- Implemented the mechanism to bypass the authentication routine during runtime.
- ASCSECOC-367 Fixed known issue: New authentic Tx PDU(s) are not being accepted in case the Tx Confirmation was not given

**Module version 2.6.0**
2020-02-21

- Implemented the SecOCSameBufferPduCollection option to link a collection of PDUs to use a buffer.

**Module version 2.5.2**
2020-01-23

- Extended the custom verification status propagation

**Module version 2.5.1**
2019-12-06

- Improved module handling by splitting source code in Rx/Tx separate files

**Module version 2.5.0**
2019-10-11

- ASCSECOC-334 Fixed known issue: Synchronous processing of the Rx PDU is interrupted when the verification result is negative

**Module version 2.4.2**
2019-09-06

- Implemented the option to propagate MAC verification return code to the application

**Module version 2.4.1**
2019-08-09

- Implemented support for RTE with FunctionElision = TRUE

**Module version 2.4.0**
2019-06-14

- Improved the SecOC state machine handling

**Module version 2.3.2**
2019-06-07

- Implemented option to propagate the MAC generate status when the service was successful or not

**Module version 2.3.1**
2019-05-17

- Improved the Csm job IDs handling
- Implemented synchronous Pdu processing for Rx and Tx side

**Module version 2.3.0**
2019-02-15

- Implemented option to skip the verification procedure by calling SecOC_VerifyStatusOverride with the overrideStatus parameter set to 43. In the case where the SecOCRxSecuredPduLayer configuration parameter is set to SecOCRxSecuredPduCollection, the lower layer authentic PDU is forwarded directly to the upper layer without waiting for the corresponding cryptographic PDU.
- Implemented the reception overflow strategies REJECT and REPLACE

**Module version 2.2.3**
2019-01-25

- ASCSECOC-301 Fixed known issue: Server call to Freshness Management SWC is incorrectly modeled for multi-partition systems
- ASCSECOC-302 Fixed known issue: Buffer overflow occurs if freshness values are smaller than 57 bits in multi-partition systems

**Module version 2.2.2**
2018-11-23

- ASCSECOC-297 Fixed known issue: Wrong compiler abstraction macro used for function parameter's pointer class
- ASCSECOC-298 Fixed known issue: Buffer overflow in case of small authenticator length

**Module version 2.2.1**
2018-10-26

- Updated the description for some of the configuration parameters and external functions

**Module version 2.2.0**
2018-09-28

- Implemented support for post build selectable
- Improved the configuration phase, when SecOCSecuredRxPduVerification is off, no Csm jof reference needs to be selected for SecOCRxAuthServiceConfigRef.
- Extended the usecases when SecOC_GetRxFreshnessAuthData() is called by the SecOC module, this function will be called if the freshness value length of the PDU is 0 bits or the length of the authentic data that needs to be send to freshness value SWC is not 0 bits

**Module version 2.1.11**
2018-07-27

- ASCSECOC-278 Fixed known issue: Out-of-bounds access if full freshness value length is not a multiple of 8 bits and truncated MAC length is smaller than one byte

**Module version 2.1.10**
2018-06-22

- Implemented the GetRxFreshnessAuthData and GetTxFreshnessTruncData functions and all the related functionality.
- ASCSECOC-271 Fixed known issue: Link error if no PDU is configured with SecOCPduType = SECOC_TPPDU
- Updated the use of exclusive areas

**Module version 2.1.9**
2018-05-25

- ASCSECOC-262 Fixed known issue: Wrong return type for function SecOC_SPduTxConfirmation
- Implemented the option to skip the configuration of SecOCFreshnessValueFuncName and SecOCSecuredPDUTransmittedFuncName when the freshness value length is equal to 0.
- Implemented support for callout functions which are indicating the SWC/CDD that the MAC Generate procedure has failed.
- Implemented support to configure an default MAC which shall be used when the MAC could not be generated
- Extended the function SecOC_VerifyStatusOverride to be able to override the Csm_MacVerify return value and callback result to "Pass".
- Implemented support for DataId length up to 32 bits.
- Implemented support for SecOCPduType SECOC_TPPDU

**Module version 2.1.8**
2018-04-20

- ASCSECOC-256 Fixed known issue: IMPLEMENTATION_CONFIG_VARIANT is not enabled
- Implemented support for PduLengthType of 32 bits
- Implemented support for secured PDU collection

**Module version 2.1.7**
2018-03-16

- Adapted the memory sections for runnable entities declared by the Rte

**Module version 2.1.6**
2018-02-16

- ASCSECOC-225 Fixed known issue: For multiple PDUs with the same SecOCFreshnessValueId, SecOC overrides status only for one PDU
- Implemented support for configuration of the Csm mode for every PDU configured in SecOC
- Implemented support for callout functions which are updating the secured PDU layout

**Module version 2.1.5**
2018-01-19

- Implemented the configuration parameter SecOCEnableForcedPassOverride and the related functionality
- Changed Primitive Implementation Data Types to Redefinition Implementation Data Types for unspecified Implementation Data Types
- Implemented the skip verification for secured PDU
- Implemented support for secured area within a Pdu

**Module version 2.1.4**
2017-12-15

- ASCSECOC-208 Fixed known issue: SecOC does not forward the PDUs to the upper layer regardless of the verification result when the configuration option SecOcIgnoreVerificationResult is enabled
- Implemented the TxConfirmation timeout
- ASCSECOC-212 Fixed known issue: If processing is not finished, the PDU length is overwritten by incoming PDU

**Module version 2.1.3**
2017-11-17

- Improved the SecOC authentication processing regarding the Tx confirmation

**Module version 2.1.2**
2017-10-20

- ASCSECOC-185 Fixed known issue: Wrong return type in SecOC function definition of Csm callback
- Improved the checking in the validation schema for the Csm jobs referenced by the SecOC module for I-PDU authentication and verification
- ASCSECOC-188 Fixed known issue: Compile error occurs if only Tx or Rx are configured with asynchronous Csm

**Module version 2.1.1**
2017-10-09

- ASCSECOC-159 Fixed known issue: Undefined macro CSM_E_VER_OK
- ASCSECOC-162 Fixed known issue: Message verification fails because the authenticator generated on Tx side is always 0
- Updated the SecOC configuration schema to AUTOSAR 4.3
- Implemented support for asynchronous Csm mode
- ASCSECOC-181 Fixed known issue: Out of bounds read access occurs if an Rx PDU has freshness length 0

**Module version 2.1.0**
2017-08-28

- ASCSECOC-106 Fixed known issue: Wrong calculation of truncated Tx freshness value bits
- Implemented support for triggered transmission

**Module version 2.0.0**
2017-08-04

- ASCSECOC-119 Fixed known issue: Inclusion of Rte_SecOC.h within SecOC.h creates compiler error
- ASCSECOC-87 Fixed known issue: Automatic calculation of PDU IDs using the Handle ID wizard does not work
- ASCSECOC-92 Fixed known issue: Authentic PDU is considered for upper layer for If and Tp module

- ASCSECOC-142 Fixed known issue: SecOC expects a Tx confirmation even if the lower layer module does not accept the transmission request
- ASCSECOC-100 Fixed known issue: SecOC does not compile if configuration parameter PduRCancelTransmit is set to false
- Implemented support for multiple secured I-PDUs with same freshness value ID
- Updated the interface operations GetRxFreshness and GetTxFreshness according to the requirement SWS_SecOC_91002 of the AUTOSAR 4.3
- Updated the type SecOC_VerificationStatusType with the element SecOCDataID according to the requirement SWS_SecOC_00160 of the AUTOSAR 4.3
- Implemented support for multiple Secured I-PDUs with the same DataIds
- ASCSECOC-133 Fixed known issue: Wrong calculation of Tx-secured PDU size for buffer clearing
- Update SecOC to use Csm synchronous single call Autosar 4.3 API

**Module version 1.2.0**
2017-04-03

- Added RfC 73691, configuration parameter SecOcIgnoreVerificationResult
- Implemented optional interface to query the freshness value from an external source (SWC or CDD)
- Implemented Bugzilla RfC 73692: Splitting the SecOC main function into an Rx- and an Tx-Path
- ASCSECOC-99 Fixed known issue: `SecOC` uses incorrect PDU ID for the Rx authentic layer PDU

**Module version 1.1.0**
2015-10-15

- Added ISO-C90 compatible interfaces for APIs SecOC_FreshnessValueWrite() and SecOC_FreshnessValueRead()

**Module version 1.0.1**
2015-06-19

- Corrected usage of the AUTOSAR memory mapping

**Module version 1.0.0**
2015-04-28

- Initial release for `SecOC` which supports a basic feature set

#### 3.3.3.2. New features

- No new features have been added since the last release.

#### 3.3.3.3. EB-specific enhancements

This chapter lists the enhancements provided by the module.

- `DataId` length up to 32 bits

  Description:

  The length for the used `DataId` can be configured for 8 bits, 16 bits or 32 bits.

  Rationale:

  The configuration of the length of the DataId allows more flexibility for the project defining the `DataIds` to be used.

- Overriding return value of MAC verification

  Description:

  The `SecOC` module provides the opportunity to override the return value of function `Csm_MacVerify` and its related callback result to "Pass" for a given number of PDUs with the same Freshness Value ID. This feature is available if SecOCEnableForcedPassOverride is set to TRUE. It is an extension to the functionality of `SecOC_VerifyStatusOverride`.

  Rationale:

  This enhancement can be used during development to authenticate PDUs also during temporary errors from the hardware used to calculate the MACs.

- Default authenticator

  Description:

  The `SecOC` module provides the configuration parameter `SecOCDefaultAuthenticatorValue`. If this parameter is enabled, and MAC generation fails, SecOC sends a secured PDU containing a default authenticator with the value defined by the configuration parameter.

  Rationale:

  This enhancement enables sending secured PDUs during development, if the generation of MACs is not available.

- Indication of MAC generation result

  Description:

Using the configuration parameter `SecOCMacGenerateStatusPropagationMode` the `SecOC` can propagate the status of the MAC generation to the application.

Rationale:

The propagation of the MAC generation result together with the feature for propagation of the MAC verification result enables the application to get all information about the current status and health of the secure onboard communication.

- Secured PDU layout callout

Description:

The `SecOC` provides the opportunity to configure callout functions which are called before sending and after receiving a secured PDU. These functions can be used to correct the bus layout for secured PDUs.

Rationale:

This enhancement can be used e.g. to add or remove padding bytes for dynamic length PDUs which require a static length on the bus (e.g. CAN-FD).

- `Csm` mode

Description:

The configuration parameter `SecOcCsmMode` allows a PDU individual configuration of the used `Csm` operation mode. It can be either ASYNCHRONOUS or SYNCHRONOUS.

Rationale:

This enhancement allows projects to optimize the processing of secured PDUs and therefore the performance and CPU load of the system.

- TxConfirmation timeout

Description:

`SecOC` provides the configuration parameter `SecOCTxConfirmationTimeout`. It can be used to define a maximum time the `SecOC` waits for a TxConfirmation from the lower layer during transmission. If no TxConfirmation was indicated until the timeout has expired, `SecOC` continues processing the next PDU.

Rationale:

This enhancement is a robustness feature to stabilize the bus communication.

- Support of *Calculate Handle IDs* wizard

Description:

The `SecOC` module supports the calculation of handle IDs with the *Calculate Handle IDs* wizard.

Rationale:

With this feature you can configure handle IDs of secured and authenticated PDUs in the SecOC module.

- Configuration parameter SecOCCryptoBitLength

Description:

The SecOC module provides the additional configuration parameter SecOCCryptoBitLength. If this parameter is enabled, the SecOC module passes the length information for the MAC in bits to the authentication service. If this parameter is disabled, the SecOC module passes the length information for the MAC in bytes to the authentication service.

Rationale:

This configuration parameter allows you to configure the SecOC module to the needs of the used cryptographic primitives.

- Configuration parameter SecOCASR403

Description:

The SecOC module provides the additional configuration parameter SecOCASR403. If this parameter is enabled, the SecOC module provides the interfaces to the PduR module as specified by AUTOSAR 4.0.3. If this parameter is disabled, the SecOC module provides the interfaces to the PduR module as specified by AUTOSAR 4.2.1.

Rationale:

This configuration parameter allows you to integrate SecOC module together with an AUTOSAR 4.0.3 PduR module as well as with an AUTOSAR 4.2.1 PduR module.

- Configuration parameter SecOCRteUsage

Description:

The SecOC module provides the additional configuration parameter SecOCRteUsage. If this parameter is enabled, the SecOC provides and uses interfaces to the Rte. If this parameter is disabled, the SecOC module does not provide or use the interfaces to the Rte. Per default SecOCRteUsage is disabled.

Rationale:

The Rte interface is not necessarily required for the usage of the SecOC module. The SecOC interface to the Rte represents a feature of the SecOC module. This feature can be used if required, but can also be disabled to reduce the code size.

### 3.3.3.4. Deviations

This chapter lists the deviations of the module from the AUTOSAR standard.

- No support of `SecOCReceptionOverflowStrategyQUEUE`

  Description:

  The option `QUEUE` for the configuration parameter `SecOCReceptionOverflowStrategy` and the functionality connected to it is not supported by the `SecOC`. Currently if the reception of a Secured I-PDU has been initiated via SecOC_StartOfReception and SecOC is busy processing a Secured I-PDU with the same PDU Identifier the SecOC can `REJECT` or `REPLACE`.

  Rationale:

  The queue mechanism of the reception strategy is an unsupported feature.

  Requirements:

  SWS_SecOC_00216, ECUC_SecOC_00076, ECUC_SecOC_00077

- No support for MetaData Handling

  Description:

  `SecOC` does not support MetaData handling. The `SecOC` module does not forward the MetaData of an authentic PDU to the corresponding secured PDU or vice versa.

  Rationale:

  MetaData handling is not sufficiently specified and only usable together with an according update of the specified reception overflow strategy as mentioned in https://bugzilla.autosar.org/show_bug.cgi?id=81343.

  Requirements:

  SWS_SecOC_00212

- No repeated transmission of authentic PDUs

  Description:

  After having successfully sent the secured PDU via triggered transmission the `SecOC` module will discard the authentic PDU and will not generate additional secured PDUs from this authentic PDU.

  Rationale:

  To reduce the ECU load the SWS_SecOC_00069 requirement is not supported by the `SecOC` module.

  Requirements:

SWS_SecOC_00069

- No support of `SecOC_ChangeParameter`

  Description:

  The `SecOC` module does not provide the ability to change a specific transport protocol parameter (e.g. block size).

  Rationale:

  The `SecOC_ChangeParameter` mechanism is not supported by the `SecOC` module.

  Requirements:

  SWS_SecOC_91011, SWS_SecOC_00218, SWS_SecOC_00103

- No support of `SecOC_CancelReceive`

  Description:

  The `SecOC` module does not provide the ability to cancel an ongoing reception of a PDU in a lower layer transport protocol module.

  Rationale:

  The `SecOC_CancelReceive` mechanism is not supported by the `SecOC` module.

  Requirements:

  SWS_SecOC_91010, SWS_SecOC_00217

- No support of development error detection

  Description:

  The `SecOC` module neither provides development errors nor calls the `Det` module. The deviation also includes all related parameters and functionalities.

  Rationale:

  The development error detection mechanism is not supported by the `SecOC` module.

  Requirements:

  SWS_SecOC_00155, SWS_SecOC_00101, SWS_SecOC_00102, SWS_SecOC_00164, SWS_SecOC_00166, ECUC_SecOC_00007, SWS_SecOC_00138, SWS_SecOC_00251, SWS_SecOC_00248

- No support of signature algorithms

  Description:

The SecOC does not provide the usage of SignatureGenerate or SignatureVerify services. Only MAC services can be used. I.e. neither signature interfaces to Csm or Cal nor signature configurations are supported by the SecOC module.

Rationale:

Authentication and verification which use signature services are not supported by the SecOC module.

Requirements:

ECUC_SecOC_00048, ECUC_SecOC_00013

- Deviation of file structure

Description:

The file structure of the SecOC module deviates from the file structure provided by the AUTOSAR specification.

- The SecOC module does not include the file Dem.h.
- Not all type definitions are defined in SecOC_Types.h. However, type definitions are available if SecOC.h is included.

Rationale:

- The SecOC module does not include the file Dem.h, because production errors are not defined.
- Not all type definitions are defined in SecOC_Types.h, because several type definitions are configuration-dependent.

Requirements:

SWS_SecOC_00002

- Secured I-PDUs can have DataId with the same value

Description:

The parameter SecOCDataId defines a numerical identifier for the Secured I-PDU. This identifier can be used for multiple Secured I-PDUs.

Rationale:

The SecOC module support multiple Secured I-PDUs with the same SecOCDataId value.

Requirements:

ECUC_SecOC_00030, ECUC_SecOC_00014

- Invalid requirement

Description:

Requirement SWS_SecOC_00049 is not feasible.

Rationale:

See https://www.autosar.org/bugzilla/show_bug.cgi?id=77622

Requirements:

SWS_SecOC_00049

- Interfaces to the PduR

Description:

The names of the interfaces to the `PduR` do not completely match the names defined in the AUTOSAR_SWS_SecureOnboardCommunication of AUTOSAR version 4.3. Because the `PduR` module is the only module to use these interfaces, this deviation has no impact usage of the `SecOC`. `SecOC` registers its interface names to the `PduR` and the `PduR` uses the provided interface names. The `SecOC` module provides and uses the `PduR` API defined by AUTOSAR version 4.0.3 or version 4.2.1 respectively.

- `SecOC_IfTxConfirmation` is named `SecOC_TxConfirmation`
- `SecOC_IfTransmit` is named `SecOC_Transmit`
- `SecOC_TpTransmit` is named `SecOC_Transmit`
- `SecOC_IfCancelTransmit` is named `SecOC_CancelTransmit`
- `SecOC_TpCancelTransmit` is named `SecOC_CancelTransmit`
- `PduR_SecOCTransmit` is named `PduR_SecOCTpTransmit` when using TP protocol
- `PduR_SecOCIfCancelTransmit` is named `PduR_SecOCCancelTransmit`
- `PduR_SecOCTpCancelTransmit` is named `PduR_SecOCCancelTransmit`
- `PduR_SecOCIfRxIndication` is named `PduR_SecOCRxIndication`

Rationale:

The interface names shall not be changed to be backward compatible.

Requirements:

SWS_SecOC_00063, SWS_SecOC_00072, SWS_SecOC_00076, SWS_SecOC_00080, SWS_SecOC_00086 SWS_SecOC_00087, SWS_SecOC_00137, SWS_SecOC_00081, SWS_SecOC_00112, SWS_SecOC_00113, SWS_SecOC_00126, SWS_SecOC_00130, SWS_SecOC_91008, SWS_SecOC_91009

- No support of Security Profiles

Description:

The SecOC does not support Security Profiles for security reasons. Nevertheless the configuration parameters of SecOC can be configured to match the given Security Profiles.

Rationale:

The risk is high, that a given Security Profile is not secure anymore in the near future, if e.g. a cryptographic algorithm is broken or a bigger key length is required to ensure security. Therefore it is highly recommended to do a security analysis on each individual use case and chose the SecOC parameters along with state of the art at the time.

Requirements:

SWS_SecOC_00190, SWS_SecOC_00191, SWS_SecOC_00192, SWS_SecOC_00193, SWS_SecOC_00194

- Names of Freshness Callout functions

Description:

The names of the callout functions SecOC_GetRxFreshness and SecOC_GetTxFreshness are not fix as defined by AUTOSAR_SWS_SecureOnboardCommunication. The names can be defined by the configuration parameters SecOCFreshnessValueFuncName.

Rationale:

The Prefix SecOC_ for a function, which is not defined by the SecOC, but another CDD or integration code violates the AUTOSAR rules.

Requirements:

SWS_SecOC_91004, SWS_SecOC_91007

- No support of configuration parameter SecOCUseTxConfirmation

Description:

The configuration parameter SecOCUseTxConfirmation is not available. It is always considered as TRUE.

Rationale:

SecOCUseTxConfirmation is an unsupported configuration parameter.

Requirements:

ECUC_SecOC_00085

- No support of configuration parameter `SecOCMaxAlignScalarType`

  Description:

  The configuration parameter `SecOCMaxAlignScalarType` is not available.

  Rationale:

  The type definition resulting from the configuration parameter `SecOCMaxAlignScalarType` is not used for `SecOC` of AUTOSAR version 4.3 and therefore the configuration parameter is obsolete.

  Requirements:

  ECUC_SecOC_00047

### 3.3.3.5. Limitations

This chapter lists the limitations of the module. Refer to the module references chapter *Integration notes*, subsection *Integration requirements* for requirements on integrating this module.

- Synchronous Pdu processing only available for synchronous Csm mode

  Description:

  Synchronous Pdu processing is not supported in case Csm is configured to execute in asynchronous mode.

  Rationale:

  Synchronous Pdu processing is blocking until the Pdu is processed completely. Thus, it shall not be available if SecOC waits for an asynchronous Csm call to return.

- SameBufferPduCollection not supported in combination with

  Description:

  The Rx same buffer PDU collection cannot be used with the Rx secured PDU collection (SecOCRxSecuredPduLayer = SecOCRxSecuredPduCollection) or with SecOCReceptionOverflowStrategy set to REPLACE.

  Rationale:

  The above combination are not supported because the reception procedure would require a strict scheduling of the different PDUs that are using the same buffer.

#### 3.3.3.6. Open-source software

SecOC does not use open-source software.

# 4. ACG8 Crypto and Security Stack user guide

## 4.1. Overview

This user guide describes the concepts and the configuration of the following modules:

- `CryIf` Crypto Interface
- `Csm` Crypto Service Manager
- `SecOC` Secure Onboard Communication

This user guide is intended for readers who have good knowledge of AUTOSAR and about the purpose of the Crypto and Security Stack modules. The information provided here helps you to integrate `CryIf`, `Csm`, and `SecOC` in an AUTOSAR project.

For instructions on how to configure the modules, see:

- Section 4.4, “CryIf module user guide”
- Section 4.5, “Csm module user guide”
- Section 4.6, “SecOC module user guide”

## 4.2. Background information

The ACG8 Crypto and Security Stack offers standardized access to cryptographic services for applications and system functions. The ACG8 Crypto and Security Stack allows you to create several configurations for various cryptographic services with individual primitives and operations.

In the following, a cryptographic functionality as provided by the ACG8 Crypto and Security Stack is called a *service*, e.g. *Encrypt* or *Hash*. The corresponding cryptographic algorithm provided by a Crypto Driver module which fulfills this cryptographic functionality is called a *primitive*, e.g. *AES-ECB encryption* or *SHA-2*.

### 4.2.1. Dependencies of the Crypto and Security Stack modules

The modules of the ACG8 Crypto and Security Stack are related as shown in Figure 4.1, “Crypto and Security Stack architecture”.

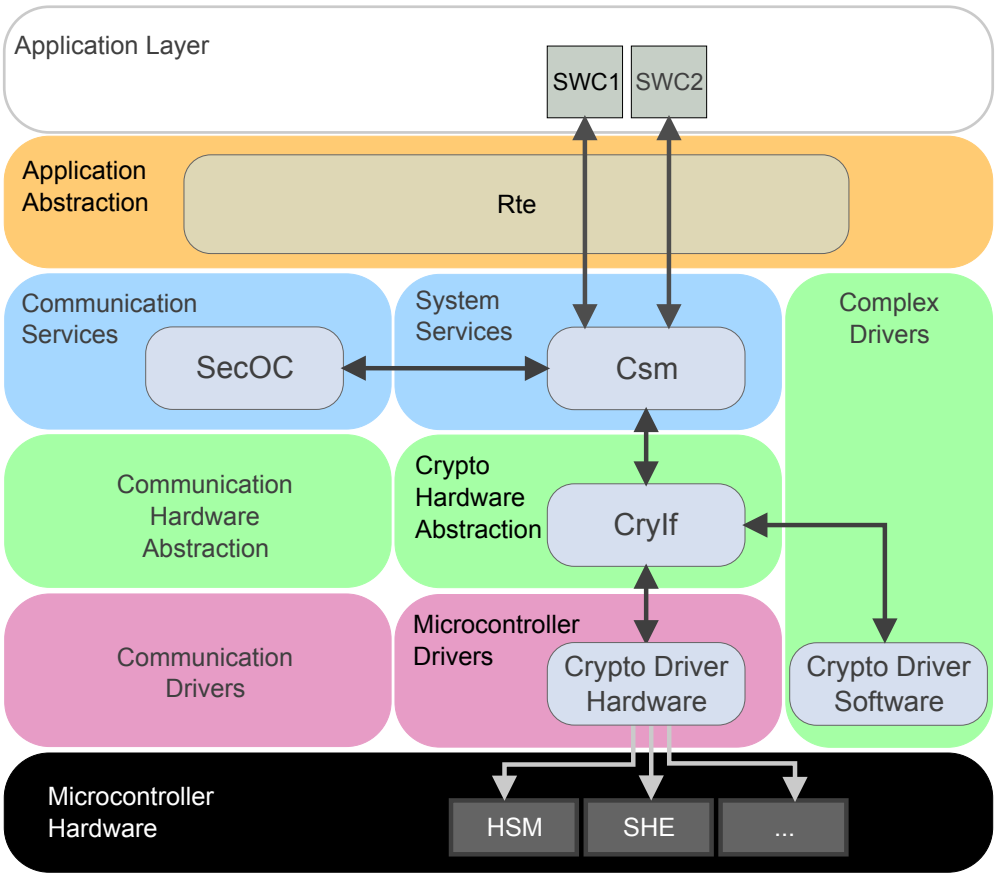


Figure 4.1. Crypto and Security Stack architecture

To provide cryptographic functionalities, an ECU needs to integrate one Crypto Service Manager module and one Crypto Interface module. The Crypto Interface module can access several Crypto Driver modules which can be realized by a software implementation or a hardware driver.

The Crypto Service Manager `Csm` offers access to cryptographic functionalities and provides a standardized interface to `Rte` and any software component (SWC) above the `Rte`, to `SecOC`, and to any software-based complex device driver (CDD). `Csm` does not perform any cryptographic functionalities itself. `Csm` sends corresponding requests to the Crypto Interface `CryIf`, which is located in the hardware abstraction layer below `Csm`.

`CryIf` forwards the `Csm` requests to the underlying crypto solutions. Crypto solutions may consist of hardware-based Crypto Drivers and/or software-based CDDs. Mixed setups with multiple Crypto Drivers are possible.

The Crypto Drivers perform the cryptographic calculations as requested by `CryIf`. `CryIf` then returns the outcome to `Csm`.

The `SecOC` module uses the cryptographic services provided by `Csm` to apply authentication mechanisms for critical data on the level of PDUs.

## 4.2.2. Secure onboard communication with MAC

This use case explains the basic configuration of the modules of the ACG8 Crypto and Security Stack to realize a secure onboard communication. When a message is sent on the bus, it might be subject to unauthorized manipulation. The secure onboard communication ensures that received data comes from the right ECU and has the correct value. To achieve this, a `SecOC` module is integrated on the level of the PDU router, both on sender and receiver side. Each `SecOC` module uses the cryptographic services provided by the `Csm`. Figure 4.2, "Modules involved in secure onboard communication" depicts the setup.

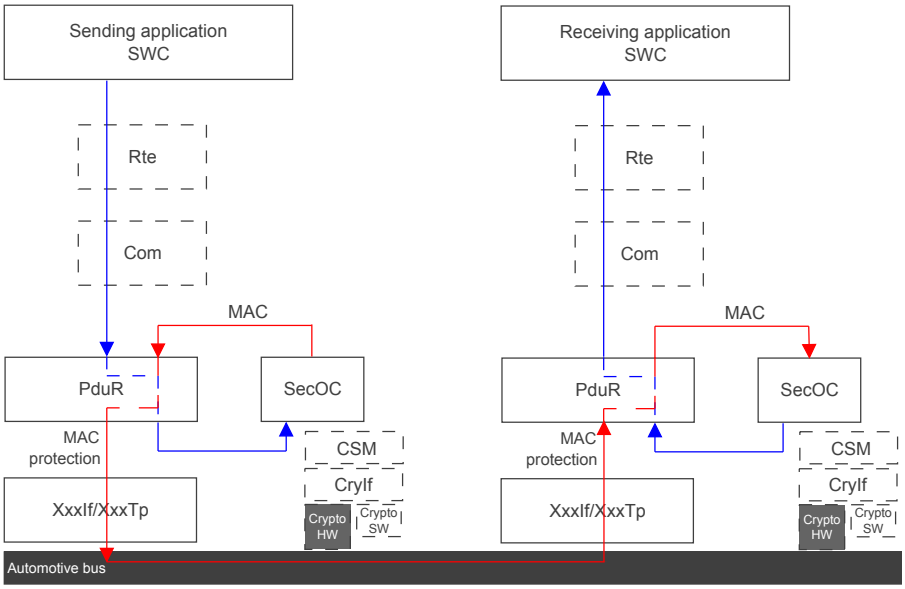


Figure 4.2. Modules involved in secure onboard communication

The `PduR` module routes incoming and outgoing security-related I-PDUs to the `SecOC` module. On the sender side, `SecOC` creates a secured I-PDU by adding a message authentication code (MAC) with a freshness value to the outgoing authentic I-PDU. The `SecOC` module on the receiver side verifies the authentication information before it passes the I-PDU to the receiver. The MAC creation and verification are managed by the `Csm`. The `Csm` uses the cryptographic algorithms of an underlying Crypto Driver for this purpose. Figure 4.3, "Interaction of ACG8 Crypto and Security Stack modules" depicts the interaction of the ACG8 Crypto and Security Stack modules for the secure onboard communication.

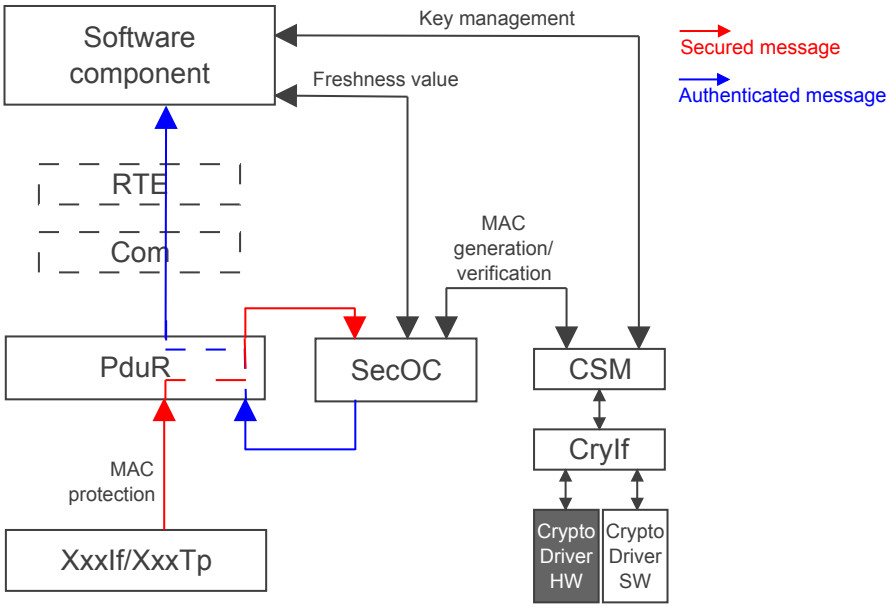


Figure 4.3. Interaction of ACG8 Crypto and Security Stack modules

## 4.2.3. Explicit and implicit restart

In the ACG8 Crypto and Security Stack, a job needs to be always restarted in an explicit way. A job is explicitly restarted as follows:

1. A job was started or is running.
2. The job is canceled via an API call.
3. The job is started again.

With regard to an implicit restart, the AUTOSAR specifications do not take a uniform approach:

- [SWS_Crypto_00020][4]: A job was started or is running. Without canceling the previous job, the same job is started again. Alternatively, the previous job was finished. This restarts the job implicitly.
- [SWS_Csm_00017][1]: An implicit restart is not specified. If a job is running, `Csm` returns BUSY for any restart request.

The ACG8 Crypto and Security Stack does not support the implicit restart. Consequently, if you configure an implicit restart of a job from within an application, the operation fails. To ensure compatibility of applications with other AUTOSAR 4.3.0 crypto stacks and for a consistent configuration of job restarts, it is recommended to use explicit restarts only.

# 4.3. Configuring the Crypto and Security Stack

To perform a certain cryptographic service, you need to configure all modules of the ACG8 Crypto and Security Stack to work together correctly. We recommend to start in the lowest layer and move your way up:

- Create a `Crypto` Driver configuration for the desired cryptographic primitive. See the documentation of the corresponding `Crypto` Driver module on how to create a valid configuration.
- Create a `CryIf` configuration for the desired cryptographic primitive and keys.
- Create a `Csm` configuration for the desired cryptographic service.
- Integrate `SecOC` module(s) according to your requirements.

Within each module, the data path configuration is separated from the key management. This separation allows you to change the crypto algorithm without modifying the data paths in the application.

To illustrate the module dependencies, the *Secure Onboard Communication* use case describes the configuration steps in the different modules of the ACG8 Crypto and Security Stack.

## 4.3.1. Configuring a secure onboard communication for an ECU

The following is a bottom-up walkthrough of the ACG8 Crypto and Security Stack modules with the basic configuration steps for a secure onboard communication. To verify messages, message authentication codes (MACs) are used. The MAC generation and verification shall be based on a symmetric-key algorithm of the family AES to generate a cipher-based message authentication code CMAC.

**Prerequisites**

- `SecOC` is registered in `PduR` as BSW module. For information on how to do this, see Section 4.6.3.1, “Registering the module in PduR”
- In `EcuC`, one authenticated and one secured global PDU exist for Tx and for Rx.
- The implemented Crypto Driver contains a Crypto Driver Object that offers the crypto primitives `MAC_Generate` and `MAC_Verify`.
- The implemented Crypto Driver contains a Crypto key that references the Crypto key type `MAC`.

Referencing the Crypto driver information in the Crypto Interface

Step 1
In `CryIf`, add a dedicated `CryIf` channel, e.g. `CryIfChannel_usecase`.

Step 2
Reference `CryIfChannel_usecase` to the Crypto driver object that contains the MAC primitives.

Step 3
Add a dedicated `CryIf` key, e.g. `CryIfKey_usecase`.

Step 4
Reference the `CryIf` key to the corresponding Crypto Driver key.

Setting up a job in the Crypto Service Manager

Step 1
In `Csm`, add a dedicated `Csm` queue, e.g. `CsmQueue_usecase`.

Step 2
Reference `CsmQueue_usecase` to `CryIfChannel_usecase`. This is the channel that you created in the Crypto Interface module.

Step 3
Create a dedicated `Csm` key, e.g. `CsmKey_usecase` and reference it to the `CryIfKey_usecase` that you created in the Crypto Interface module.

Step 4
For `CsmKey_usecase`, enable the **Use port** checkbox. An Rte port is required because the use case involves a software component (SWC) that receives messages or handles freshness values and keys for `Csm`. As the SWC is located in the upper layer, it communicates with `Csm` via the Rte.

Step 5
Configure the required service primitives. The ECU can both send and receive messages, so it needs a service primitive that generates a MAC as well as a service primitive that verifies a MAC.

Step 5.1
For MAC generation, enable the `CsmMacGenerate` primitive. Set the desired algorithm family and algorithm mode, e.g. algorithm family `AES` with algorithm mode `CMAC`, and the processing to `synchronous` or `asynchronous`.

Step 5.2
For MAC verification, enable the `CsmMacVerify` primitive. Set the desired algorithm family and algorithm mode, e.g. to algorithm family `AES` with algorithm mode `CMAC`, and the processing to `synchronous` or `asynchronous`.

Step 6
Create two Csm jobs: a `MacGenerate` job and a `MacVerify` job. For each, reference the `CsmKey_usecase` and the `Csm` primitives for MAC generation/MAC verification that you configured.

Step 7
Disable the **Use port** checkbox. These `Csm` jobs are addressed to the `SecOC` module, which is like `Csm` a BSW module located below the Rte.

Step 8
Reference your dedicated `CsmQueue`. Both jobs can reference the same queue.

Specifying the messages to be verified and linking them to the Csm job

Step 1
In `SecOC`, configure the path for reception and verification of a message, i.e. specify which secured PDU shall be verified by the `SecOC` module.

Step 1.1
On the **Secured RX Pdus** tab, the table shows all global Rx PDUs that were configured in `EcuC`. Select a `SecOCRxPduProcessing` entry.

Step 1.2
For the `SecOCRxSecuredLayerPduRef` parameter, reference the secured PDU.

Step 1.3
For the `SecOCRxAuthenticLayerPduRef` parameter, reference the corresponding authenticated PDU.

Step 1.4
In **AuthAlgorithm**, for the `SecOCRxAuthServiceConfigRef` parameter, select `CsmJob_MacVerify`. This is the job you configured in `Csm` for MAC verification.

Step 2
To configure the path for the secured PDU, proceed accordingly:

Step 2.1
On the **Secured TX Pdus** tab, the table shows all global Tx PDUs that were configured in `EcuC`. Select a `SecOCTxPduProcessing` entry.

Step 2.2
For the `SecOCTxSecuredLayerPduRef` parameter, reference the secured PDU.

Step 2.3
For the `SecOCTxAuthenticLayerPduRef` parameter, reference the corresponding authenticated PDU.

Step 2.4
In **AuthAlgorithm**, for the `SecOCTxAuthServiceConfigRef` parameter, select `CsmJob_MacGenerate`. This is the job you configured in `Csm` for MAC generation.

You completed the basic configuration of the modules involved in a secure onboard communication use case. For further configuration details, see the user guides of the individual modules.

# 4.4. Crylf module user guide

## 4.4.1. Overview

This chapter provides `CryIf` specific information:

- Section 4.4.2, “Background information” explains the basic functionality of `CryIf`.
- Section 4.4.3, “Configuring the CryIf module” provides configuration information.

For `CryIf` parameter descriptions, see Chapter 5, “ACG8 Crypto and Security Stack module references“.

## 4.4.2. Background information

Located between the lower level `Crypto` driver module and the `Csm` in the upper service layer, `CryIf` provides a unique and standardized interface to manage different cryptographic services. `CryIf` maintains a mapping scheme which allows `Csm` to use multiple `Crypto` hardware and software solutions. `CryIf` does not perform any cryptographic calculations itself.

It receives cryptographic service requests from the `Csm`. `CryIf` then calls the corresponding `Crypto` driver to perform the cryptographic calculations.

`CryIf` is the only user of the `Crypto` drivers and ensures concurrent access to them. Thus, multiple crypto tasks can be processed at the same time.

## 4.4.3. Configuring the CryIf module

`CryIf` can be seen as a routing layer. For this purpose, you need to create `CryIf` channels that link a `Csm` queue to a specific `Crypto` driver object. Furthermore, you create specific `CryIf` keys which reference the desired key of a `Crypto` driver module.

The result is a unique key mapping with regard to all your existing `Crypto` driver modules. The mapping order is arbitrary. Also, you can create fewer keys in `CryIf` than your `Crypto` driver modules offer. With the mapping in `CryIf` you define what is to be communicated to the upper layer.

Configuring the CryIf module

Prerequisite:

- You created a `Crypto` driver configuration for the desired cryptographic primitives and keys. At least the following elements are configured: `CryptoKey`, `CryptoDriverObject`. See the documentation of the corresponding `Crypto` driver module on how to create a valid configuration.

Step 1
On the **CryIfChannels** tab, enter a name for the new `CryIf` channel.

Step 2
For the `CryIfChannelId` parameter, enter an ID number for the `CryIf` channel.

Step 3
For the `CryIfDriverObjectRef` parameter, select the `Crypto` driver object to which the `Csm` queue is to be connected.

Step 4
To create a `CryIf` key, on the **CryIfKeys** tab, enter a name for the new key.

Step 5
For the `CryIfKeyId` parameter, enter a ID number for the `CryIf` key.

Step 6
For the `CryIfKeyRef` parameter, reference the desired key from the `Crypto` driver module.

Step 7
[optional]

If the *module implementation prefixes*, i.e. `vendorId` and `vendorApiInfix`, of multiple Crypto driver modules shall be determined based on their BSWMDs, do the following:

Step 7.1
Generate the BSWMDs for all desired `Crypto` driver modules.

Step 7.2
Import the generated BSWMDs for all desired `Crypto` driver modules.

Step 7.3
Enable the `CryIfEbGeneralBswmdImplementation` container.

Step 7.4
Add an entry per desired `Crypto` driver module.

Step 7.5
For the `CryIfCryptoRef` parameter, select the desired `Crypto` driver module per entry.

Step 7.6
For the `CryIfCryptoBswImplementationRef` parameter, select the corresponding reference for the desired `Crypto` driver module per entry.

**TIP**

**Copying Crypto driver keys**

In `CryIf`, you can copy keys from one `Crypto` driver module to another. A key is copied element by element. `CryIf` needs an internal buffer for this task. With the `CryIfGeneral/CryIfMaxKeyElementCopySize` parameter, you can configure the size of this internal buffer to reduce the memory usage.

If the configured size is less than the size of a key element in the source key, the copying fails unless partial access is enabled for this key element. If partial access is enabled for a key element in the source key, the copied bytes are limited by the configured size `CryIfMaxKeyElementCopySize`. If partial access is enabled for a key element in the target key, and partial data with a size less than the source key element size was written to the key element, the copying fails.

To prevent this, ensure that the current key element sizes are large enough for the corresponding source key elements. You can do this by writing data with at least the needed size to the affected key elements.

# 4.5. Csm module user guide

## 4.5.1. Overview

This chapter provides `Csm` specific information:

- Section 4.5.2, “Background information” explains the basic functionality.
- Section 4.5.3, “Configuring the Csm module” provides configuration information.

For `Csm` parameter descriptions, see Chapter 5, “ACG8 Crypto and Security Stack module references“.

## 4.5.2. Background information

With the Crypto Service Manager `Csm`, you manage the various cryptographic service requests from the different applications, from `SecOC`, and possible CDDs. The `Csm` allows for different service requests to use the same service but with different underlying primitives. For example, one application might use the *Hash* service to compute a SHA-2 algorithm while another application might use a SHA-3. Also, `Csm` can process multiple independent jobs in parallel.

The `Csm` uses the following terms related to cryptographic functionality:

- **Service:** the capability of a cryptographic primitive (e.g. AEAD_DECRYPT, AEAD_ENCRYPT, ENCRYPT, DECRYPT, HASH, MAC_GENERATE, MAC_VERIFY, RANDOM)

- **Family:** the algorithm family of a primitive (e.g. 3DES, AES, RSA, SHA2_256, ED25519)
- **Mode:** the algorithm mode of a primitive (e.g. ECB, CBC, CMAC, RSASSA_PSS)
- **Csm Primitive:** the combination of a service, a family and a mode

### 4.5.3. Configuring the Csm module

To manage the various crypto service requests from the applications, you need to configure specific `Csm` jobs. A `Csm` job is a combination of a cryptographic key, a job queue, a priority and a description of the desired cryptographic primitive that is to supposed to be executed by a `Crypto` driver module. The key and the job queue are related to an individual `Crypto` driver object of a `Crypto` module, which is accessed via the `CryIf` module. The priority defines how immediate the job is executed. The higher the value you enter for the priority, the more immediate the job is executed.

You can set up a `Csm` job for synchronous or asynchronous processing.

`Csm` does not offer a preconfiguration because the settings depend on your configurations in the lower levels.

Configuring the Csm module

Prerequisite:

- A `CryIf` configuration must exist at least for: `CryIfChannel`, `CryIfKey`.

Step 1
On the **CsmQueue** tab, enter a name for the new `Csm` queue.

Step 2
For the `CsmChannelRef` parameter, reference the channel that you configured in `CryIf/CryIfChannels`.

Step 3
For the `CsmQueueSize` parameter, enter the number of requests that this queue should be able to process.

Step 4
On the **CsmKey** tab, enter a name for the new `Csm` key.

Step 5
For the `CsmKeyId` parameter, enter an ID for the `Csm` key. The IDs must be in ascending order without gaps, starting from 0.

Step 6
For the `CsmKeyRef` parameter, reference the key that you configured in `CryIf/CryIfKeys`.

Step 7
Enable the **CsmKeyUsePort** checkbox for this key if an RTE service interface or port is required for reading and writing the key. As a general rule, this is the case if `Csm` is to be used by a software component. If the

`Csm` is to interact with another BSW module, you do not need to enable this checkbox because the modules communicate via API, below the RTE layer.

Step 8
On the **CsmCallback** tab, configure a callback function. A callback informs about the end of a job if you select asynchronous job processing. Depending on your project, you can configure a general callback or define specific callback functions for different scenarios.

Step 9
On the **CsmPrimitives** tab, enable the desired service and configure the underlying primitive as follows:

Step 9.1
Select the algorithm family from the drop-down list.

Step 9.2
Select the algorithm mode from the drop-down list.

Step 9.3
For the `Csm<service>Processing` parameter, decide whether this `Csm` service should be executed synchronously or asynchronously. For details, see Section 4.5.3.1, “Synchronous or asynchronous job processing”.

**NOTE**

**Only one active service**

Make sure that you enable exactly one service per primitive at a time. It does not work if you do not enable a service at all or if you enable multiple services for the same primitive.

A primitive often has a counterpart: encrypt-decrypt, generate-verify. Ensure that you configure both primitives in such a case. An example of a primitive without a counterpart is the RandomGenerate service.

Step 10
On the **CsmJob** tab, reference the `CsmKey`, the `CsmPrimitive`, and the `CsmQueue` to create a `Csm` job.

Step 11
Enable the checkbox **CsmJobUsePort** if an RTE port is required to execute this job.

### 4.5.3.1. Synchronous or asynchronous job processing

You can configure all services to be processed synchronously or asynchronously with the configuration parameter `Csm<service>Processing`. If a service supports the streaming approach, the calling application can either use the streaming mode or a single-call. The streaming mode comprises the streaming call sequence *Start, Update, Finish*.

The desired mode is selected in the interface's `mode` parameter. The calling application passes the required mode via `Crypto_OperationModeType` to the interface's `mode` parameter.

If the streaming approach is used, the call sequence of the modes must be as follows:

1. CRYPTO_OPERATIONMODE_START
2. CRYPTO_OPERATIONMODE_UPDATE
3. CRYPTO_OPERATIONMODE_FINISH

CRYPTO_OPERATIONMODE_UPDATE can be called multiple times. The modes CRYPTO_OPERATIONMODE_START and CRYPTO_OPERATIONMODE_UPDATE can be combined to a single call with the mode CRYPTO_OPERATIONMODE_STREAMSTART.

The call with the next mode can only be performed if the previous call returned the positive value `E_OK`. In case of asynchronous job processing, the configured callback and the callback result need to be awaited before the next job is processed.

With mode CRYPTO_OPERATIONMODE_SINGLECALL, the functionality with all calculations is calculated with one call to the service.

For synchronous job processing, the functionality result is available when the function returns with the positive return value `E_OK`.

For asynchronous job processing, the functionality result is available when the callback function is invoked with the positive return value `CSM_E_OK`.

# 4.6. SecOC module user guide

## 4.6.1. Overview

This chapter provides `SecOC` specific information:

- Section 4.6.2, “Background information” explains the basic functionality.
- Section 4.6.3, “Configuring SecOC” provides configuration information.

For `SecOC` parameter descriptions, see Chapter 5, “ACG8 Crypto and Security Stack module references“.

## 4.6.2. Background information

The `SecOC` module provides functionality to verify the authenticity and freshness of PDU-based communication between ECUs within the vehicle architecture. The approach requires both the sending ECU and the receiving

ECU to implement a SecOC module. The SecOC module is integrated with the upper and lower layer PduR APIs on the sender and receiver side. The SecOC module interacts with the PduR module.

On the sender side, the SecOC module creates a secured I-PDU by adding authentication information to the outgoing authentic I-PDU. Figure 4.4, “Layout of a secured I-PDU” shows the layout of a secured I-PDU. The authentication information is comprised of an authenticator and a freshness value. An authenticator is e.g. a message authentication code (MAC). The authenticator is computed from the freshness value and the authentic I-PDU. The freshness value is obtained from a freshness manager on sender and receiver side. The freshness manager can be a software component or a complex driver. On the receiver side, the SecOC module checks the freshness and authenticity of the authentic I-PDU by verifying the authentication information that was appended by the sending side SecOC module.

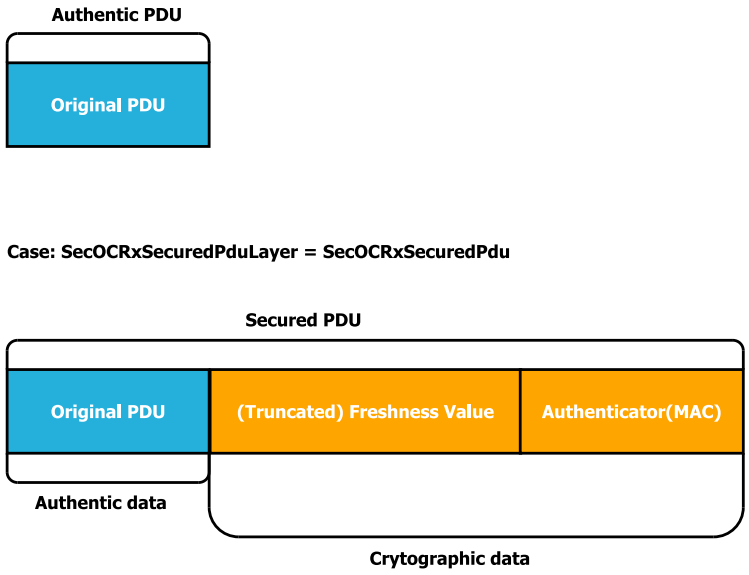


Figure 4.4. Layout of a secured I-PDU

The SecOC module is also able to send and receive the secured I-PDU in the form of two individual PDUs, the authentic PDU and the cryptographic PDU as depicted in Figure 4.5, “Layout of secured PDU collection”

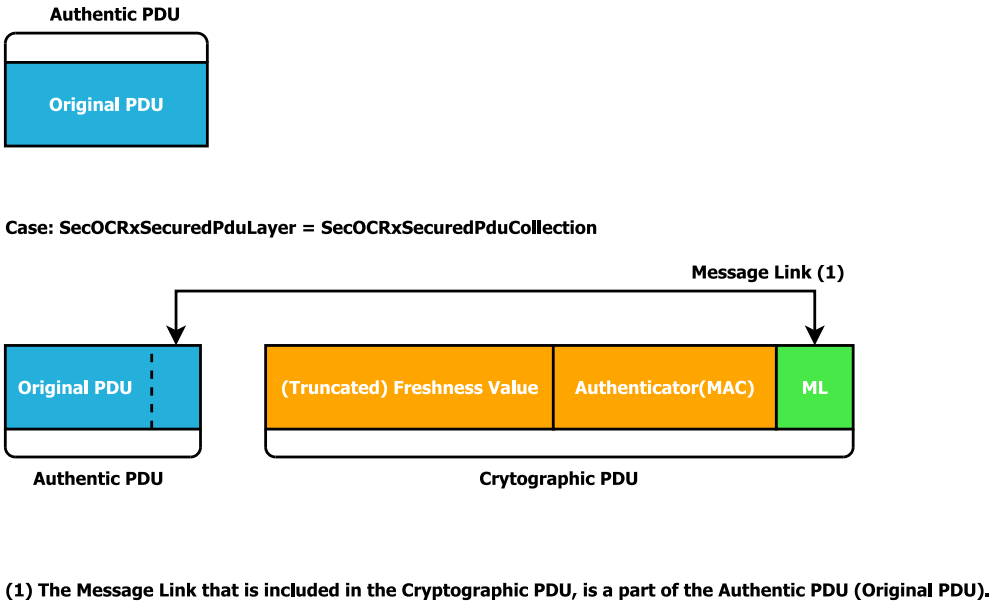


Figure 4.5. Layout of secured PDU collection

The `SecOC` module uses cryptographic services of the `Crypto Service Manager` (`Csm`) module to calculate the authenticator. The `SecOC` module uses MAC generation on sender side and MAC verification on receiver side.

## 4.6.3. Configuring SecOC

This section describes how to configure the `SecOC` module into a EB tresos Studio project. The scope of the description here is for projects where the module needs to be configured manually. You may skip this section if the system description of the project contains the configuration of the `SecOC` module.

### 4.6.3.1. Registering the module in PduR

Registering the module in PduR

Step 1
Open the editor of the `PduR` module and open the tab **PduRBswModules**.

Step 2
Add a new entry to the row and name it `SecOC`.

Step 3
Enable the following parameters:

- PduRLowerModule
- PduRTxConfirmation
- PduRBswModuleIsEnabled
- PduRCalculateHandleId
- PduRCommunicationInterface and/or PduRTransportProtocol
- PduRUpperModule
- PduRUseTag is enabled by default and cannot be edited.

#### 4.6.3.2. Configuring Rte dependencies

The `SecOC` module includes a service software component. The related interfaces are defined in the file `SecOC_swcd_interfaces.arxml`. You can use this file to develop software components that interact with the `SecOC` module.

#### 4.6.3.3. Configuring the Tx path

For each secured transmit I-PDU, you need the following objects from surrounding modules. If they do not exist already, you need to create them.

- `EcuC` module
  - Global PDU for the authentic I-PDU
  - Global PDU for the secured I-PDU
    - Length = `<length of authentic I-PDU> + <length of SecOCFreshnessValueTxLength> + <length of SecOCAuthInfoTxLength>`
- `PduR` module

**NOTE** **Routing path references**

Both routing paths need to refer to their respective global PDUs. One routing path refers to one common global PDU for its source and destination.

  - I-PDU routing path for the authentic I-PDU
  - I-PDU routing path for the secured I-PDU
- `Csm` module
  - Configuration of a `Csm` job which references a `CsmMacGenerate` service
- `<Net>If` or `<Net>Tp` module which depends on the network, e.g. CAN, FlexRay, Ethernet

- I-PDU to send the secured I-PDU
- `Com` or `Dcm` module
  - The authentic I-PDU
- `SecOC`
  - Select the tab **Secured TX Pdus** and add an entry for every Tx PDU which shall be sent secured by the `SecOC` module.

**TIP**

**Configuring handle IDs for the SecOC module automatically**

Use the *Calculate Handle IDs* wizard to automatically configure the handle IDs for the `SecOC` PDUs in all related modules. For more information about the Calculate Handle IDs wizard, see the user guide of EB tresos Studio.

### 4.6.3.4. Configuring the Rx path

For each secured received I-PDU, you need the following objects from surrounding modules. If they do not exist already, you need to create them.

- `EcuC` module
  - Global PDU for the authentic I-PDU
  - Global PDU for the secured I-PDU
    - Length = `<length of authentic I-PDU> + <length of SecOCFreshnessValueTxLength> + <length of SecOCAuthInfoTxLength>`
- `PduR` module

**NOTE**

**Routing path references**

Both routing paths need to refer to their respective global PDUs. One routing path refers to one common global PDU for its source and destination.

  - I-PDU routing path for the authentic I-PDU
  - I-PDU routing path for the secured I-PDU
- `Csm` module
  - Configuration of a `Csm` job which references a `CsmMacVerify` service
- `<Net>If` or `<Net>Tp` module which depends on the network, e.g. CAN, FlexRay, Ethernet
  - I-PDU to receive the secured I-PDU
- `Com` or `Dcm` module

- I-PDU which contains the received authentic I-PDU

- SecOC
  - Select the tab **Secured RX Pdus** and add an entry for every Rx PDU which shall be received secured by the SecOC module.

**TIP**

**Configuring handle IDs for the SecOC module automatically**

Use the *Calculate Handle IDs* wizard to automatically configure the handle IDs for the SecOC PDUs in all related modules. For more information about the Calculate Handle IDs wizard, see the user guide of EB tresos Studio.

### 4.6.3.5. Selecting the communication interface

- The SecOC module supports direct transmission as well as transport protocol transmission. For each PDU, you can select the transmission to be used with the configuration parameter **SecOCPduType**. The values are SECOC_TPPDU for transport protocol or SECOC_IFPDU for direct transmission.
- If in the PduR configuration described in Section 4.6.3.1, “Registering the module in PduR” the parameter **PduRTriggertransmit** is enabled, SecOC also supports triggered transmission.
- Every Tx PDU configured according to Section 4.6.3.3, “Configuring the Tx path” has a configuration parameter **SecOCTxConfirmationTimeout** which defines the time that SecOC waits for a Tx confirmation from the lower layer.

### 4.6.3.6. Defining the layout of a secured PDU

SecOC provides several options to define the layout of a secured PDU. The secured PDU depicted in Figure 4.4, “Layout of a secured I-PDU” is the standard layout.

Defining the secured PDU layout

Step 1
In the tab **Secured TX Pdus** or **Secured RX Pdus** respectively, select an entry for a secured PDU.

Step 2
Define the parameters **SecOCTxSecuredPduLayer** and **SecOCRxSecuredPduLayer**: For Tx PDUs, select either SecOCTxSecuredPdu or SecOCTxSecuredPduCollection. For Rx PDUs, select either SecOCRxSecuredPdu or SecOCRxSecuredPduCollection.

SecOCTxSecuredPdu and SecOCRxSecuredPdu specify the layout of a secured PDU as depicted in figure Figure 4.4, “Layout of a secured I-PDU”. That means there is one secured PDU per authentic PDU.

Step 2.1
In **SecOCTxSecuredLayerPduRef** and **SecOCRxSecuredLayerPduRef** respectively, reference a global secured PDU.

The global secured PDU shall have at least the length defined in Section 4.6.3.3, “Configuring the Tx path” and Section 4.6.3.4, “Configuring the Rx path”

For `SecOCTxSecuredPduCollection` and `SecOCRxSecuredPduCollection`, the secured PDU consists of two separate PDUs which are sent or received on the bus: an authenticated PDU and a cryptographic PDU as depicted in Figure 4.5, “Layout of secured PDU collection”

The authenticated PDU contains only the authentic data. The cryptographic PDU contains the authentication information and an optional message linker.

Step 2.1
Instead of one secured PDU per authentic PDU, define two global PDUs in the `EcuC` module: an authenticated and a cryptographic PDU.

Step 2.2
Configure the size of the authenticated PDU with the same size as the authentic PDU that holds the data to be secured.

Step 2.3
Configure the length of the cryptographic PDU with a value as follows: `<length of SecOCFreshnessValueTxLength> + <length of SecOCAuthInfoTxLength> + <length of SecOCMessageLinkLen>`.

Step 2.4
For **SecOCTxAuthenticPduRef** in the **SecOCTxSecuredPduCollection** and for **SecOCRxAuthenticPduRef** in the **SecOCRxSecuredPduCollection**, reference the authenticated global PDU of the `EcuC`.

Step 2.5
For **SecOCTxCryptographicPduRef** in the **SecOCTxSecuredPduCollection** and for **SecOCRxCryptographicPduRef** in the **SecOCRxSecuredPduCollection**, reference the cryptographic global PDU of the `EcuC`.

Step 2.6
Optionally, enable the container **SecOCUseMessageLink**.

A message linker is a part of the authentic PDU. It is added to the cryptographic PDU on sender side. On receiver side, it is used to determine whether a pair of received authenticated and cryptographic PDU belongs together, before performing any cryptographic calculations.

- In **SecOCMessageLinkLen**, you define the length of the message linker.
- In **SecOCMessageLinkPos**, you define the start position of the data within the authentic PDU which is used as message linker in the cryptographic PDU.

Step 3
Define a secured area.

If the parameter **SecOCUseSecuredArea** and the containers **SecOCTxPduSecuredArea** and **SecOCRxPduSecuredArea** are disabled, the complete authentic PDU is subject to the MAC calculations for the secured PDU's authenticator.

If only a part of the authentic PDU shall be relevant to the secured PDU's authenticator, perform the following steps:

Step 3.1
Enable the parameter **SecOCUseSecuredArea**.

Step 3.2
Enable the container **SecOCTxPduSecuredArea** or **SecOCRxPduSecuredArea** respectively.

Step 3.3
Define the length **SecOCSecuredTxPduLength** or **SecOCSecuredRxPduLength** of the data, which shall be taken into account for cryptographic calculations.

Step 3.4
Define the offset **SecOCSecuredTxPduOffset** or **SecOCSecuredRxPduOffset** within the authentic PDU for the data, which shall be taken into account for cryptographic calculations.

Step 4
The secured PDUs created by the `SecOC` are bus-independent and are compatible to all common communication protocols.

In special cases, the created secured PDU might not fit some constraints of the used bus. For example, if dynamic length PDUs shall be secured by `SecOC` and sent with a CAN-FD bus, it might be necessary to add padding bytes to the secured PDU. These padding bytes are specific to both project and bus. Also other project specific deviations from the AUTOSAR defined secured PDU layout can be considered. Therefore, they need to be handled by a callout function.

`SecOC` calls this function right before handing over the secured Tx PDU to the `PduR` module or right after obtaining the secured Rx PDU from the `PduR`.

To use such a callout function:

Step 4.1
Enable the configuration parameters **SecOCTxShapeFuncName** or **SecOCRxShapeFuncName** respectively in the `SecOC` **General** tab and enter the name of the C-function which implements the callout.

Step 4.2
Enable the configuration parameters **SecOCTxUseShapeFunc** or **SecOCRxUseShapeFunc** respectively for the relevant PDUs.

The following interfaces shall be available for `SecOC` when:

- the parameter `SecOCRxShapeFuncName` is configured:

```
Std_ReturnType 'SecOCRxShapeFuncName' ( PduIdType SecOCPduID, uint8* SecPdu,
PduLengthType* SrcSecPduLength, const PduLengthType* DstSecPduLength, uint32
AuthenticatorLength )
```

- the parameter `SecOCTxShapeFuncName` is configured:

  ```
  Std_ReturnType 'SecOCTxShapeFuncName' ( PduIdType SecOCPduID, uint8* SecPdu,
  const PduLengthType* SrcSecPduLength, PduLengthType* DstSecPduLength, uint32
  AuthenticatorLength )
  ```

### 4.6.3.7. Configuring the freshness

With the `SecOCQueryFreshnessValue` parameter, you configure the creation of freshness values. The following options are available:

- `RTE`: A freshness value is called from a SWC for every PDU that uses an `RTE` service port. `SecOC` creates a require service port to obtain freshness values for Rx verification and a require service port to obtain freshness values for Tx authentication. These ports shall be connected to the corresponding provide service ports of a SWC, which is providing the freshness values to `SecOC`. For Tx PDUs, `SecOC` calls the operation `SPduTxConfirmation` when a secured PDU was transmitted.

  The following interfaces shall be available for `SecOC` when at least one Rx PDU is configured:

  - if the `SecOCUseAuthDataFreshness` is enabled:

    ```
    Std_ReturnType 'SwcName'_GetRxFreshnessAuthData ( uint16 SecOCFreshness-
    ValueID, SecOC_FreshnessArrayType* SecOCTruncatedFreshnessValue, uint32
    SecOCTruncatedFreshnessValueLength, SecOC_FreshnessArrayType* SecOCAu-
    thDataFreshnessValue, uint16 SecOCAuthDataFreshnessValueLength, uint16
    SecOCAuthVerifyAttempts, SecOC_FreshnessArrayType* SecOCFreshnessValue,
    uint32* SecOCFreshnessValueLength )
    ```

  - if the `SecOCUseAuthDataFreshness` is disabled:

    ```
    Std_ReturnType 'SwcName'_GetRxFreshness ( uint16 SecOCFreshnessValueID, Se-
    cOC_FreshnessArrayType* SecOCTruncatedFreshnessValue, uint32 SecOCTruncat-
    edFreshnessValueLength, uint16 SecOCCounterSyncAttempts, SecOC_Freshnes-
    sArrayType* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength )
    ```

  The following interfaces shall be available for `SecOC` when at least one Tx PDU is configured:

  - if the `SecOCProvideTxTruncatedFreshnessValue` is enabled:

    ```
    Std_ReturnType 'SwcName'_GetTxFreshnessTruncData ( uint16 SecOCFresh-
    nessValueID, SecOC_FreshnessArrayType* SecOCFreshnessValue, uint32* Se-
    cOCFreshnessValueLength SecOC_FreshnessArrayType* SecOCTruncatedFreshness-
    Value, uint32* SecOCTruncatedFreshnessValueLength )
    ```

  - if the `SecOCProvideTxTruncatedFreshnessValue` is disabled:

```
Std_ReturnType 'SwcName'_GetTxFreshness ( uint16 SecOCFreshnessValueID, SecOC_FreshnessArrayType* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength )
```

- `Std_ReturnType 'SwcName'_SPduTxConfirmation (uint16 SecOCFreshnessValueID)`

- `CFUNC`: A freshness value is queried from a CDD for every PDU using a C-function. For each `SecOCFreshnessValueId`, it is possible to define a function name using the configuration parameter `SecOCFreshnessValueFuncName`. For Tx PDUs, `SecOC` calls the operation `SPduTxConfirmation` when a secured PDU was transmitted.

The following interfaces shall be available for `SecOC` when at least one Rx PDU is configured:

- if the `SecOCUseAuthDataFreshness` is enabled:

```
Std_ReturnType 'SecOCFreshnessValueFuncNameRx_UseAuthDataFreshness' ( uint16 SecOCFreshnessValueID, uint8* SecOCTruncatedFreshnessValue, uint32 SecOCTruncatedFreshnessValueLength, uint8* SecOCAuthDataFreshnessValue, uint16 SecOCAuthDataFreshnessValueLength, uint16 SecOCAuthVerifyAttempts, uint8* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength )
```

- if the `SecOCUseAuthDataFreshness` is disabled:

```
Std_ReturnType 'SecOCFreshnessValueFuncNameRx' ( uint16 SecOCFreshnessValueID, uint8* SecOCTruncatedFreshnessValue, uint32 SecOCTruncatedFreshnessValueLength, uint16 SecOCCounterSyncAttempts, uint8* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength )
```

The following interfaces shall be available for `SecOC` when at least one Tx PDU is configured:

- if the `SecOCProvideTxTruncatedFreshnessValue` is enabled:

```
Std_ReturnType 'SecOCFreshnessValueFuncNameTx_TruncatedFreshnessValue' ( uint16 SecOCFreshnessValueID, uint8 *SecOCFreshnessValue, uint32 *SecOCFreshnessValueLength uint8 *SecOCTruncatedFreshnessValue, uint32 *SecOCTruncatedFreshnessValueLength )
```

- if the `SecOCProvideTxTruncatedFreshnessValue` is disabled:

```
Std_ReturnType 'SecOCFreshnessValueFuncNameTx' ( uint16 SecOCFreshnessValueID, uint8* SecOCFreshnessValue, uint32* SecOCFreshnessValueLength )
```

- `void 'SecOCSecuredPDUTransmittedFuncName' (uint16 SecOCFreshnessValueID)`

- `NONE`: The freshness value mechanism will not be used.

### 4.6.3.8. Configuring authenticator verification

The SecOC secures authentic data with an authenticator. The authenticator consists of a cryptographic MAC calculated for the authentic data and a freshness value using a cryptographic key. The authenticator is generated on sender side and verified on receiver side. The SecOC module utilizes the Csm module for cryptographic calculations.

Configuring authenticator information

Prerequisite:

- You configured the Csm, the CryIf and the Crypto modules as described in Section 4.3.1, “Configuring a secure onboard communication for an ECU”.

Step 1
Configure the authenticator generation for Tx PDUs:

Step 1.1
On the **Secured TX Pdus** tab, the table shows all Tx PDUs that were configured. Select a SecOCTxPduProcessing entry.

Step 1.2
In **AuthAlgorithm**, for the SecOCTxAuthServiceConfigRef parameter, select the job you configured in Csm for MAC generation.

Step 1.3
The Csm MAC generation can be executed synchronously or asynchronously. Select the desired mode with configuration parameter SecOcCsmMode.

Step 1.4
Either the complete MAC or only the most significant bits of the authenticator are included in the secured PDU on the bus. In configuration parameter SecOCAuthInfoTxLength, add the length of the authenticator which shall be part of the secured PDU.

Step 1.5
The MAC calculations might not be successful with the first call to the Csm functions. For example, the job queue of the crypto stack might be full. Define the number of attempts for the MAC calculation requests with the configuration parameter SecOCAuthenticationBuildAttempts.

Step 1.6
Set the SecOCMacGenerateStatusPropagationMode to FAILURE_ONLY if the SecOC shall notify the application about a failure of the MAC generation.

To propagate the MAC generation status to the application, ensure that the respective Rte port is connected or that at least one C-function is configured in the list of the **MacGenerateStatus Callout** tab.

Step 1.7
During development phase, the cryptographic functionality might not be available and therefore the MAC generation might fail. In this case, `SecOC` drops the PDU without forwarding it when the maximum number of build attempts as configured in `SecOCAuthenticationBuildAttempts` is reached.

To create and send a secured PDU despite a MAC generation failure, switch to the **General** tab, enable the configuration parameter `SecOCDefaultAuthenticatorValue` and set the value to the MAC value pattern to be used for the secured PDU.

Step 2
Configure the authenticator verification for Rx PDUs:

Step 2.1
On the **Secured RX Pdus** tab, the table shows all Rx PDUs that were configured. Select a `SecOCRxPduProcessing` entry.

Step 2.2
In **AuthAlgorithm**, for the `SecOCRxAuthServiceConfigRef` parameter, select the job you configured in `Csm` for MAC verification.

Step 2.3
The `Csm` MAC verification can be executed synchronously or asynchronously. Select the desired mode with configuration parameter `SecOcCsmMode`.

Step 2.4
Either the complete MAC or only the most significant bits of the authenticator are included in the secured PDU on the bus. In configuration parameter `SecOCAuthInfoTxLength`, add the length of the authenticator which is part of the secured PDU.

Step 2.5
The MAC calculations might not be successful with the first call to the `Csm` functions. For example, the job queue of the crypto stack might be full. Define the number of attempts for the MAC calculation requests with the configuration parameter `SecOCAuthenticationBuildAttempts`.

Step 2.6
The freshness value used for MAC calculations might be not completely contained in the secured PDU on the bus. Therefore, the receiver side has to reconstruct the freshness value for MAC verification. If the reconstructed freshness value is not equal to the value used for the MAC generation on sender side, the MAC verification fails. In configuration parameter `SecOCAuthenticationVerifyAttempts`, define the number of retries for reconstructing and verifying the freshness value.

Step 2.7
Set the `SecOCVerificationStatusPropagationMode` to BOTH or FAILURE_ONLY if the `SecOC` shall notify the application about the MAC verification result.

To propagate the MAC verification status to the application, ensure that the parameter `SecOCPropagateVerificationStatus` is enabled and the respective `Rte` port is connected or that at least one C-function is configured in the list of the **VerificationStatus Callout** tab.

Step 2.8
If the verification of a secured PDU is not successful, `SecOC` drops the message and does not forward it to the upper layer.

To ignore the verification result of all secured PDUs during development phase and pass the authentic data to the upper layer, enable the configuration parameter `SecOcIgnoreVerificationResult` in the **General** tab.

Step 2.9
With configuration parameter `SecOCSecuredRxPduVerification` of every `SecOCRxPduProcessing` entry, you can define individually for each PDU whether the verification shall be performed or skipped.

Step 2.10
To control the verification result in certain use cases during runtime, you can use the interface `VerifyStatusOverride` to overwrite the verification result of a PDU with FAIL. If it shall also be possible to override the status with PASS, enable the configuration parameter `SecOCEnableForcedPassOverride` in the **General** tab.

### 4.6.3.9. Configuring required RAM for Post-Build usage

The amount of RAM for Post-Build usage that is required for `SecOC` needs to be calculated and set. This can be done with the configuration parameters `SecOCMaxPduBufferSize` and `SecOCMaxIntBufferSize`.

Configuring required RAM for Post-Build usage

Prerequisite:

- The `SecOC` configuration must be performed entirely.
- Rationale: to be able to calculate the amount of RAM the number of Rx and Tx PDUs, the layout of the secured PDU must be configured in order to obtain the required values.


Figure 4.6. Calculate required RAM

**NOTE**

**Calculate the required RAM when multiple variants are configured**

When multiple post build variants are configured for `SecOC`, Step 1 and Step 2 must be performed for every variant and the biggest value must be set for the parameters `SecOC-MaxPduBufferSize` and `SecOCMaxIntBufferSize`.

Step 1

Define the value for **Max buffer size for PDUs** by using the **Calculate value** button presented in Figure 4.6, “Calculate required RAM”.

Step 2

Define the value for **Max buffer size for internal usage** by using the **Calculate value** button presented in Figure 4.6, “Calculate required RAM”.

# 5. ACG8 Crypto and Security Stack module references

## 5.1. Overview

This chapter provides module references for the ACG8 Crypto and Security Stack product modules. These include a detailed description of all configuration parameters. Furthermore this chapter lists the application programming interface with all data types, constants and functions.

The content of the sections is sorted alphabetically according the EB tresos AutoCore Generic module names.

For further information on the functional behavior of these modules, refer to the chapter ACG8 Crypto and Security Stack user's guide.

### 5.1.1. Notation in EB module references

EB notation may differ from the AUTOSAR standard notation in the software specification documents (SWS). This section describes the notation of *default value* and *range* fields in the EB module references.

#### 5.1.1.1. Default value of configuration parameters

If there is no default value specified for a parameter, the default value field is omitted to prevent ambiguity with parameters that have `--` as default values.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has no default value field, therefore it is omitted.

#### 5.1.1.2. Range information of configuration parameters

The range of a configuration parameter contains an upper and a lower boundary. However, in special cases the range of allowed values can be computed by means of an XPath function that is evaluated at configuration time. An XPath function can either be a standard `xpath:<function>()` or a custom `cxpath:<function>()` function. The range of a configuration parameter may be computed based on other configuration parameters that are referenced from the XPath function. For more information on custom XPath functions, see section *Custom XPath Functions API* of the EB tresos Studio developer's guide.

Example: The parameter `BswMCompuConstText` of the `BswM` module of EB tresos AutoCore Generic 8 Mode Management has the custom XPath function `cxpath:getCompuMethodsVT()` in the range field which provides the allowed values.

# 5.2. CryIf

## 5.2.1. Configuration parameters

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CommonPublishedInformation | 1..1 | **Label:** Common Published Information<br>Common container, aggregated by all modules. It contains published information about vendor and versions. |
| CryIfGeneral | 1..1 | **Label:** CryIfGeneral<br><br>Container for incorporation of CryIfGeneral. |
| CryIfChannel | 0..n | **Label:** CryIfChannel<br><br>Container for incorporation of CryIfChannel. |
| CryIfKey | 0..n | **Label:** CryIfKey<br><br>Container for incorporation of CryIfKey. |
| CryIfEbGeneral | 1..1 | Container for EB specific common configurations. |
| PublishedInformation | 1..1 | **Label:** EB Published Information<br>Additional published parameters not covered by CommonPublishedInformation container. |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| IMPLEMENTATION_CONFIG_VARIANT | 1..1 |

| **Parameter Name** | **IMPLEMENTATION_CONFIG_VARIANT** |
|---|---|
| **Label** | Config Variant |
| **Description** | Select the configuration variant. Currently only PreCompile is supported. |
| **Multiplicity** | 1..1 |

| | |
|---|---|
| **Type** | ENUMERATION |
| **Default value** | VariantPreCompile |
| **Range** | VariantPreCompile |

#### 5.2.1.1. CommonPublishedInformation

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| ArMajorVersion | 1..1 |
| ArMinorVersion | 1..1 |
| ArPatchVersion | 1..1 |
| SwMajorVersion | 1..1 |
| SwMinorVersion | 1..1 |
| SwPatchVersion | 1..1 |
| ModuleId | 1..1 |
| VendorId | 1..1 |
| Release | 1..1 |

| **Parameter Name** | **ArMajorVersion** |
|---|---|
| **Label** | AUTOSAR Major Version |
| **Description** | Major version number of AUTOSAR specification on which the appropriate implementation is based on. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 4 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **ArMinorVersion** |
|---|---|
| **Label** | AUTOSAR Minor Version |
| **Description** | Minor version number of AUTOSAR specification on which the appropriate implementation is based on. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |

| | |
|---|---|
| **Default value** | 3 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **ArPatchVersion** |
|---|---|
| **Label** | AUTOSAR Patch Version |
| **Description** | Patch level version number of AUTOSAR specification on which the appropriate implementation is based on. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 0 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwMajorVersion** |
|---|---|
| **Label** | Software Major Version |
| **Description** | Major version number of the vendor specific implementation of the module. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 1 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwMinorVersion** |
|---|---|
| **Label** | Software Minor Version |
| **Description** | Minor version number of the vendor specific implementation of the module. The numbering is vendor specific. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 0 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwPatchVersion** |
|---|---|

| | |
|---|---|
| **Label** | Software Patch Version |
| **Description** | Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 27 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **ModuleId** |
|---|---|
| **Label** | Numeric Module ID |
| **Description** | Module ID of this module from Module List |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 112 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **VendorId** |
|---|---|
| **Label** | Vendor ID |
| **Description** | Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 1 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **Release** |
|---|---|
| **Label** | Release Information |
| **Multiplicity** | 1..1 |
| **Type** | STRING_LABEL |
| **Default value** | |
| **Configuration class** | **PublishedInformation:** |

| | |
|---|---|
| **Origin** | Elektrobit Automotive GmbH |

### 5.2.1.2. CryIfGeneral

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CryIfDevErrorDetect | 1..1 |
| CryIfMaxKeyElementCopySize | 1..1 |
| CryIfVersionInfoApi | 1..1 |

| **Parameter Name** | **CryIfDevErrorDetect** | |
|---|---|---|
| **Label** | CryIfDevErrorDetect | |
| **Description** | Switches the development error detection and notification on or off.<br>► TRUE = detection and notification is enabled<br>► FALSE = detection and notification is disabled | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CryIfMaxKeyElementCopySize** |
|---|---|
| **Label** | CryIfMaxKeyElementCopySize |
| **Description** | The maximum buffer size in bytes used for copy processes of key elements between different Crypto drivers. This buffer is not used for copy processes of key elements within the same Crypto driver.<br>Range:<br>► Integer : 1 .. "the size of the largest key element of all referenced keys in 'CryIfKey'" |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1 |

| | |
|---|---|
| | <=node:fallback("->num:i(num:max(node:refs(node:refs(node:refs(as:modconf('CryIf')/CryIfKey/*/CryIfKeyRef)/CryptoKeyTypeRef)/CryptoKeyElementRef/*)/CryptoKeyElementSize \| 1))", "4294967295") |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit |

| **Parameter Name** | **CryIfVersionInfoApi** |
|---|---|
| **Label** | CryIfVersionInfoApi |
| **Description** | Pre-processor switch to enable and disable availability of the API CryIf_GetVersionInfo().<br>► TRUE = API CryIf_GetVersionInfo() is available<br>► FALSE = API CryIf_GetVersionInfo() is not available |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |
| **Default value** | false |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.2.1.3. CryIfChannel

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CryIfChannelId | 1..1 |
| CryIfDriverObjectRef | 1..1 |

| **Parameter Name** | **CryIfChannelId** |
|---|---|
| **Label** | CryIfChannelId |
| **Description** | Identifier of the crypto channel.<br>Specifies to which crypto channel the CSM queue is connected to.<br>Range:<br>► Integer : 0 .. 4294967295 |
| **Multiplicity** | 1..1 |

| | | |
|---|---|---|
| **Type** | INTEGER | |
| **Range** | >=0<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CryIfDriverObjectRef** | |
|---|---|---|
| **Label** | CryIfDriverObjectRef | |
| **Description** | This parameter refers to a Crypto Driver Object.<br><br>Specifies to which Crypto Driver Object the crypto channel is connected to. | |
| **Multiplicity** | 1..1 | |
| **Type** | SYMBOLIC-NAME-REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.2.1.4. CryIfKey

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CryIfKeyId | 1..1 |
| CryIfKeyRef | 1..1 |

| **Parameter Name** | **CryIfKeyId** |
|---|---|
| **Label** | CryIfKeyId |
| **Description** | Identifier of the CryIf key.<br><br>Specifies to which CryIf key the CSM key is mapped to.<br><br>Range:<br><br>► Integer : 0 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Range** | >=0<br><=4294967295 |

| | | |
|---|---|---|
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CryIfKeyRef** |
|---|---|
| **Label** | CryIfKeyRef |
| **Description** | This parameter refers to the crypto driver key.<br><br>Specifies to which crypto driver key the CryIf key is mapped to. |
| **Multiplicity** | 1..1 |
| **Type** | SYMBOLIC-NAME-REFERENCE |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.2.1.5. CryIfEbGeneral

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CryIfEbMisc | 1..1 | Configuration of miscellaneous options. |
| CryIfEbGeneralBswmdImplementation | 0..1 | Container for configuring multiple Crypto modules to be used by the CryIf via driver APIs using the vendorId and vendorApiInfix of a specific driver as specified in its BSWMD.<br><br>► DISABLED = vendorId and vendorApiInfix of all Crypto modules are determined via CommonPublishedInformation.<br><br>► ENABLED = vendorId and vendorApiInfix of configured Crypto drivers are determined via BSWMD and for not configured Crypto drivers via CommonPublishedInformation. |

### 5.2.1.6. CryIfEbMisc

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CryIfEbAutosarApiVersion | 1..1 |

| **Parameter Name** | **CryIfEbAutosarApiVersion** |
|---|---|

| | |
|---|---|
| **Description** | Switches the compatibility of the CryIf module API and ARXML description as specified by the configured AUTOSAR version.<br>► CRYIF_API_VERSION_430 = Provide and expect an API and ARXML description as specified by AUTOSAR v4.3.0. Deviations are documented in the release notes.<br>► CRYIF_API_VERSION_431 = Provide and expect an API and ARXML description as specified by AUTOSAR v4.3.1. Deviations are documented in the release notes.<br>► CRYIF_API_VERSION_EB = Provide and expect an API and ARXML description as used by EB in conjunction with Csm modules less than version 3.1.0 and Crypto modules less than version 2.0.0. |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYIF_API_VERSION_430 |
| **Range** | CRYIF_API_VERSION_430<br>CRYIF_API_VERSION_431<br>CRYIF_API_VERSION_EB |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit Automotive GmbH |

### 5.2.1.7. CryIfEbGeneralBswmdImplementation

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CryIfEbGeneralBswmdImplementationRefs | 1..n | **Label:** CryIfEbGeneralBswmdReferences<br>Container to configure a specific Crypto module whose vendorId and vendorApiInfix shall be determined from its BSWMD. |

### 5.2.1.8. CryIfEbGeneralBswmdImplementationRefs

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |

| Parameters included | |
|---|---|
| CryIfCryptoRef | 1..1 |
| CryIfCryptoBswImplementationRef | 1..1 |

| Parameter Name | CryIfCryptoRef | |
|---|---|---|
| Label | CryIfCryptoRef | |
| Description | Refers to the underlying Crypto module. | |
| Multiplicity | 1..1 | |
| Type | SYMBOLIC-NAME-REFERENCE | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | Elektrobit Automotive GmbH | |

| Parameter Name | CryIfCryptoBswImplementationRef | |
|---|---|---|
| Label | CryIfCryptoBswImplementationRef | |
| Description | Reference to the BswImplementation of the underlying driver which contains the vendorId and vendorApiInfix. | |
| Multiplicity | 1..1 | |
| Type | FOREIGN-REFERENCE | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | Elektrobit Automotive GmbH | |

### 5.2.1.9. PublishedInformation

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| PbcfgMSupport | 1..1 |

| Parameter Name | PbcfgMSupport |
|---|---|
| Label | PbcfgM support |
| Description | Specifies whether or not the CryIf can use the PbcfgM module for post-build support. |
| Multiplicity | 1..1 |
| Type | BOOLEAN |
| Default value | false |

| | |
|---|---|
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

## 5.2.2. Application programming interface (API)

### 5.2.2.1. Type definitions

#### 5.2.2.1.1. CryIf_CancelJobPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_CancelJob. |
| **Type** | `Std_ReturnType(*)(uint32, Crypto_JobInfoType *Crypto_JobType *)` |

#### 5.2.2.1.2. CryIf_CertificateParsePtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_CertificateParse. |
| **Type** | `Std_ReturnType(*)(uint32)` |

#### 5.2.2.1.3. CryIf_CertificateVerifyPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_CertificateVerify. |
| **Type** | `Std_ReturnType(*)(uint32, uint32, Crypto_VerifyResultType *)` |

#### 5.2.2.1.4. CryIf_KeyCopyPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyCopy. |
| **Type** | `Std_ReturnType(*)(uint32, uint32)` |

#### 5.2.2.1.5. CryIf_KeyDerivePtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyDerive. |

| | |
|---|---|
| **Type** | `Std_ReturnType(*)(uint32, uint32)` |

#### 5.2.2.1.6. CryIf_KeyElementCopyPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyElementCopy. |
| **Type** | `Std_ReturnType(*)(uint32, uint32, uint32, uint32)` |

#### 5.2.2.1.7. CryIf_KeyElementGetPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyElementGet. |
| **Type** | `Std_ReturnType(*)(uint32, uint32, uint8 *, uint32 *)` |

#### 5.2.2.1.8. CryIf_KeyElementIdsPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyElementIds. |
| **Type** | `Std_ReturnType(*)(uint32, uint32 *, uint32 *)` |

#### 5.2.2.1.9. CryIf_KeyElementSetPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyElementSet. |
| **Type** | `Std_ReturnType(*)(uint32, uint32, const uint8 *, uint32)` |

#### 5.2.2.1.10. CryIf_KeyExchangeCalcPubValPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyExchangeCalcPubVal. |
| **Type** | `Std_ReturnType(*)(uint32, uint8 *, uint32 *)` |

#### 5.2.2.1.11. CryIf_KeyExchangeCalcSecretPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyExchangeCalcSecret. |
| **Type** | `Std_ReturnType(*)(uint32, const uint8 *partnerPublicValuePtr, uint32)` |

#### 5.2.2.1.12. CryIf_KeyGeneratePtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeyGenerate. |
| **Type** | `Std_ReturnType(*)(uint32)` |

#### 5.2.2.1.13. CryIf_KeySetValidPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_KeySetValid. |
| **Type** | `Std_ReturnType(*)(uint32)` |

#### 5.2.2.1.14. CryIf_ProcessJobPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_ProcessJob. |
| **Type** | `Std_ReturnType(*)(uint32, Crypto_JobType *)` |

#### 5.2.2.1.15. CryIf_RandomSeedPtrType

| | |
|---|---|
| **Purpose** | Function pointer type for CryIf_RandomSeed. |
| **Type** | `Std_ReturnType(*)(uint32, const uint8 *, uint32)` |

### 5.2.2.2. Macro constants

#### 5.2.2.2.1. CRYIF_CHANNEL_COUNT

| | |
|---|---|
| **Purpose** | Number of cryif channels. |
| **Value** | {number of configured CryIf channels} |

#### 5.2.2.2.2. CRYIF_CHANNEL_xxChannelIdxx_CRY_CHANNEL_ID

| | |
|---|---|
| **Purpose** | CryIf Channel. |
| **Value** | {Id} |

#### 5.2.2.2.3. CRYIF_DEV_ERROR_DETECT

| | |
|---|---|
| **Purpose** | Configuration parameter CryIfDevErrorDetection. |
| **Value** | STD_ON or STD_OFF |

#### 5.2.2.2.4. CRYIF_E_INIT_FAILED

| | |
|---|---|
| **Purpose** | Error Code for init failed. |
| **Value** | 0x01U |

#### 5.2.2.2.5. CRYIF_E_KEY_SIZE_MISMATCH

| | |
|---|---|
| **Purpose** | Error Code for key size mismatch. |
| **Value** | 0x05U |

#### 5.2.2.2.6. CRYIF_E_PARAM_HANDLE

| | |
|---|---|
| **Purpose** | Error Code for invalid handle. |
| **Value** | 0x03U |

#### 5.2.2.2.7. CRYIF_E_PARAM_POINTER

| | |
|---|---|
| **Purpose** | Error Code for invalid pointer. |
| **Value** | 0x02U |

#### 5.2.2.2.8. CRYIF_E_PARAM_VALUE

| | |
|---|---|
| **Purpose** | Error Code for invalid value. |
| **Value** | 0x04U |

#### 5.2.2.2.9. CRYIF_E_UNINIT

| | |
|---|---|
| **Purpose** | Error Code for uninitialized module. |
| **Value** | 0x00U |

#### 5.2.2.2.10. CRYIF_INSTANCE_ID

| | |
|---|---|
| **Purpose** | Instance ID of the Crypto Interface. |
| **Value** | 0x00U |

#### 5.2.2.2.11. CRYIF_KEY_COUNT

| | |
|---|---|
| **Purpose** | Number of cryif keys. |
| **Value** | {number of configured CryIf keys} |

#### 5.2.2.2.12. CRYIF_KEY_xxCryIfKeyIdxx_CRY_KEY_ID

| | |
|---|---|
| **Purpose** | CryIf Key. |
| **Value** | {Id} |

#### 5.2.2.2.13. CRYIF_MAX_KEY_ELEMNT_COPY_SIZE

| | |
|---|---|
| **Purpose** | Maximum key size for key or element copy in bytes. |
| **Value** | {size} |

#### 5.2.2.2.14. CRYIF_SID_CALLBACKNOTIFICATION

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_CallbackNotification. |
| **Value** | 0x0DU |

#### 5.2.2.2.15. CRYIF_SID_CANCELJOB

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_CancelJob. |
| **Value** | 0x0EU |

#### 5.2.2.2.16. CRYIF_SID_CERTIFICATEPARSE

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_CertificateParse. |
| **Value** | 0x0CU |

#### 5.2.2.2.17. CRYIF_SID_CERTIFICATEVERIFY

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_CertificateVerify. |
| **Value** | 0x11U |

#### 5.2.2.2.18. CRYIF_SID_GETVERSIONINFO

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_GetVersionInfo. |
| **Value** | 0x01U |

#### 5.2.2.2.19. CRYIF_SID_INIT

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_Init. |
| **Value** | 0x00U |

#### 5.2.2.2.20. CRYIF_SID_KEYCOPY

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyCopy. |
| **Value** | 0x10U |

#### 5.2.2.2.21. CRYIF_SID_KEYDERIVE

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyDerive. |
| **Value** | 0x09U |

#### 5.2.2.2.22. CRYIF_SID_KEYELEMENTCOPY

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyElementCopy. |
| **Value** | 0x0FU |

#### 5.2.2.2.23. CRYIF_SID_KEYELEMENTGET

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyElementGet. |
| **Value** | 0x06U |

#### 5.2.2.2.24. CRYIF_SID_KEYELEMENTSET

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyElementSet. |
| **Value** | 0x04U |

#### 5.2.2.2.25. CRYIF_SID_KEYEXCHANGECALCPUBVAL

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyExchangeCalcPubVal. |
| **Value** | 0x0AU |

#### 5.2.2.2.26. CRYIF_SID_KEYEXCHANGECALCSECRET

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyExchangeCalcSecret. |
| **Value** | 0x0BU |

#### 5.2.2.2.27. CRYIF_SID_KEYGENERATE

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeyGenerate. |
| **Value** | 0x08U |

#### 5.2.2.2.28. CRYIF_SID_KEYSETVALID

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_KeySetValid. |
| **Value** | 0x05U |

#### 5.2.2.2.29. CRYIF_SID_PROCESSJOB

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_ProcessJob. |
| **Value** | 0x03U |

#### 5.2.2.2.30. CRYIF_SID_RANDOMSEED

| | |
|---|---|
| **Purpose** | AUTOSAR API service ID for CryIf_RandomSeed. |
| **Value** | 0x07U |

#### 5.2.2.2.31. CRYIF_VERSION_INFO_API

| | |
|---|---|
| **Purpose** | Configuration parameter CryIfVersionInfoApi. |
| **Value** | STD_ON or STD_OFF |

### 5.2.2.3. Objects

#### 5.2.2.3.1. CryIf_CancelJobJumpTable

| | |
|---|---|
| **Purpose** | CancelJob Jumptable for different Crypto Driver Objects. |
| Type | `const CryIf_CancelJobPtrType` |

#### 5.2.2.3.2. CryIf_CertificateParseJumpTable

| | |
|---|---|
| **Purpose** | CertificateParse Jumptable for different Crypto Driver Objects. |
| Type | `const CryIf_CertificateParsePtrType` |

#### 5.2.2.3.3. CryIf_CertificateVerifyJumpTable

| | |
|---|---|
| **Purpose** | CertificateVerify Jumptable for different Crypto Driver Objects. |
| Type | `const CryIf_CertificateVerifyPtrType` |

#### 5.2.2.3.4. CryIf_Channels

| | |
|---|---|
| **Purpose** | Container for the Crypto Channels. |
| Type | `const uint32` |

#### 5.2.2.3.5. CryIf_KeyCopyJumpTable

| | |
|---|---|
| **Purpose** | KeyCopy Jumptable for different Crypto Drivers. |
| Type | `const CryIf_KeyCopyPtrType` |

#### 5.2.2.3.6. CryIf_KeyDeriveJumpTable

| **Purpose** | KeyDerive Jumptable for different Crypto Driver Objects. |
|---|---|
| Type | `const CryIf_KeyDerivePtrType` |

#### 5.2.2.3.7. CryIf_KeyElementCopyJumpTable

| **Purpose** | keyElementCopy Jumptable for different Crypto Driver Objects |
|---|---|
| Type | `const CryIf_KeyElementCopyPtrType` |

#### 5.2.2.3.8. CryIf_KeyElementGetJumpTable

| **Purpose** | keyElementGet Jumptable for different Crypto Driver Objects |
|---|---|
| Type | `const CryIf_KeyElementGetPtrType` |

#### 5.2.2.3.9. CryIf_KeyElementIdsGetJumpTable

| **Purpose** | keyElementsIdGet Jumptable for different Crypto Driver Objects |
|---|---|
| Type | `const CryIf_KeyElementIdsPtrType` |

#### 5.2.2.3.10. CryIf_KeyElementSetJumpTable

| **Purpose** | keyElementSet Jumptable for different Crypto Driver Objects |
|---|---|
| Type | `const CryIf_KeyElementSetPtrType` |

#### 5.2.2.3.11. CryIf_KeyExchangeCalcPubValJumpTable

| **Purpose** | KeyExchangeCalcPubVal Jumptable for different Crypto Driver Objects. |
|---|---|
| Type | `const CryIf_KeyExchangeCalcPubValPtrType` |

#### 5.2.2.3.12. CryIf_KeyExchangeCalcSecretJumpTable

| **Purpose** | KeyExchangeCalcSecret Jumptable for different Crypto Driver Objects. |
|---|---|
| Type | `const CryIf_KeyExchangeCalcSecretPtrType` |

#### 5.2.2.3.13. CryIf_KeyGenerateJumpTable

| **Purpose** | KeyGenerate Jumptable for different Crypto Driver Objects. |
|---|---|
| Type | `const CryIf_KeyGeneratePtrType` |

#### 5.2.2.3.14. CryIf_KeySetValidJumpTable

| **Purpose** | keySetValid Jumptable for different Crypto Driver Objects |
|---|---|
| Type | `const CryIf_KeySetValidPtrType` |

#### 5.2.2.3.15. CryIf_Keys

| **Purpose** | Container to map Crypto Interface Keys to Crypto Driver Keys. |
|---|---|
| Type | `const uint32` |

#### 5.2.2.3.16. CryIf_ProcessJobJumpTable

| **Purpose** | ProcessJob Jumptable for different Crypto Driver Objects. |
|---|---|
| Type | `const CryIf_ProcessJobPtrType` |

#### 5.2.2.3.17. CryIf_RandomSeedJumpTable

| **Purpose** | KeyGenerate Jumptable for different Crypto Driver Objects. |
|---|---|
| Type | `const CryIf_RandomSeedPtrType` |

### 5.2.2.4. Functions

#### 5.2.2.4.1. CryIf_CallbackNotification

| **Purpose** | Notifies the CryIf about the completion of the request with the result of the cryptographic operation. |
|---|---|
| **Synopsis** | `void CryIf_CallbackNotification ( const Crypto_JobType * job , Std_ReturnType result );` |
| **Service ID** | CRYIF_SID_CALLBACKNOTIFICATION |

| | | |
|---|---|---|
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (in)** | `job` | Holds a pointer to the job structure |
| | `result` | Contains the result of the cryptographic operation |

#### 5.2.2.4.2. CryIf_CancelJob

| | | |
|---|---|---|
| **Purpose** | This interface dispatches the job cancellation function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_CancelJob`** `( uint32 channelId , Crypto_JobType * job );` | |
| **Service ID** | CRYIF_SID_CANCELJOB | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `channelId` | Holds the identifier of the crypto channel. |
| **Parameters (in,out)** | `job` | Pointer to the configuration of the job. Contains structures with user and primitive relevant information. |
| **Return Value** | Standard Return Value extended by the Crypto Stack | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |

#### 5.2.2.4.3. CryIf_CertificateParse

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the certificate parse function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_CertificateParse`** `( uint32 cryIfKeyId );` | |
| **Service ID** | CRYIF_SID_CERTIFICATEPARSE | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key which shall be parsed. |
| **Return Value** | Standard Return Value | |

| | | |
|---|---|---|
| | E_OK | Request successful |
| | E_NOT_OK | Request failed |
| | E_BUSY | Request Failed, Crypto Driver Object is Busy |

#### 5.2.2.4.4. CryIf_CertificateVerify

| | | |
|---|---|---|
| **Purpose** | Verifies the certificate stored in the key referenced by verifyCryIfKeyId with the certificate stored in the key referenced by cryIfKeyId. | |
| **Synopsis** | `Std_ReturnType CryIf_CertificateVerify ( uint32 cryIfKeyId , uint32 verifyCryIfKeyId , Crypto_VerifyResultType * verifyPtr );` | |
| **Service ID** | CRYIF_SID_CERTIFICATEVERIFY | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key which shall be parsed. |
| | `verifyCryIfKeyId` | Holds the identifier of the key containing the certificate to be verified. |
| **Parameters (out)** | `verifyPtr` | Holds a pointer to the memory location which will contain the result of the certificate verification. |
| **Return Value** | Standard Return Value | |
| | E_OK | Request successful |
| | E_NOT_OK | Request failed |
| | E_BUSY | Request Failed, Crypto Driver Object is Busy |
| **Description** | {Reentrant, but not for the same cryIfKeyId} | |

#### 5.2.2.4.5. CryIf_GetVersionInfo

| | |
|---|---|
| **Purpose** | Provides information about the version of the module. |
| **Synopsis** | `void CryIf_GetVersionInfo ( Std_VersionInfoType * versioninfo );` |
| **Service ID** | CRYIF_SID_GETVERSIONINFO |
| **Sync/Async** | Synchronous |

| | | |
|---|---|---|
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `versioninfo` | Pointer to a version info structure |
| **Parameters (in,out)** | `versioninfo` | Pointer to a version info structure |

#### 5.2.2.4.6. CryIf_Init

| | |
|---|---|
| **Purpose** | Initializes the Crypto Interface module. |
| **Synopsis** | `void` **`CryIf_Init`** `( void );` |
| **Service ID** | CRYIF_SID_INIT |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |

#### 5.2.2.4.7. CryIf_KeyCopy

| | | |
|---|---|---|
| **Purpose** | This function shall copy all key elements from the source key to a target key. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyCopy`** `( uint32 cryIfKeyId , uint32 targetCryIfKeyId );` | |
| **Service ID** | CRYIF_SID_KEYCOPY | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key whose key element shall be the source element. |
| | `targetCryIfKeyId` | Holds the identifier of the key whose key element shall be the destination element. |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| | `CRYPTO_E_BUSY` | Request failed, Crypto Driver Object is busy |
| | `CRYPTO_E_KEY_EXTRACT_DENIED` | Request failed, not allowed to extract key element |
| | `CRYPTO_E_KEY_READ_FAIL` | Request failed, not allowed to extract key element |
| | `CRYPTO_E_KEY_WRITE_FAIL` | Request failed, not allowed to write key element. |

| | | |
|---|---|---|
| | CRYPTO_E_KEY_SIZE_MISMATCH | Request failed, key element sizes are not compatible. |
| **Description** | {Reentrant, but not for the same cryIfKeyId} | |

#### 5.2.2.4.8. CryIf_KeyDerive

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the key derive function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyDerive`** `( uint32 cryIfKeyId , uint32 targetCryIfKeyId );` | |
| **Service ID** | CRYIF_SID_KEYDERIVE | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key which is used for key derivation. |
| | `targetCryIfKeyId` | Holds the identifier of the key which is used to store the derived key. |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |

#### 5.2.2.4.9. CryIf_KeyElementCopy

| | | |
|---|---|---|
| **Purpose** | This function shall copy a key elements from one key to a target key. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyElementCopy`** `( uint32 cryIfKeyId , uint32 keyElementId , uint32 targetCryIfKeyId , uint32 targetKeyElementId );` | |
| **Service ID** | CRYIF_SID_KEYELEMENTCOPY | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key whose key element shall be the source element. |
| | `keyElementId` | Holds the identifier of the key element which shall be the source for the copy operation. |
| | `targetCryIfKeyId` | Holds the identifier of the key whose key element shall be the destination element. |

| | | |
|---|---|---|
| | `targetKeyElementId` | Holds the identifier of the key element which shall be the destination for the copy operation. |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| | `CRYPTO_E_BUSY` | Request failed, Crypto Driver Object is busy |
| | `CRYPTO_E_KEY_EXTRACT_DENIED` | Request failed, not allowed to extract key element |
| | `CRYPTO_E_KEY_READ_FAIL` | Request failed, not allowed to extract key element |
| | `CRYPTO_E_KEY_WRITE_FAIL` | Request failed, not allowed to write key element. |
| | `CRYPTO_E_KEY_SIZE_MISMATCH` | Request failed, key element sizes are not compatible. |
| **Description** | {Reentrant, but not for the same cryIfKeyId} | |

#### 5.2.2.4.10. CryIf_KeyElementGet

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the set key element function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyElementGet`** `( uint32 cryIfKeyId , uint32 keyElementId , uint8 * resultPtr , uint32 * resultLengthPtr );` | |
| **Service ID** | CRYIF_SID_KEYELEMENTGET | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key whose key element shall be returned. |
| | `keyElementId` | Holds the identifier of the key element which shall be returned. |
| **Parameters (in,out)** | `resultLengthPtr` | Holds a pointer to a memory location in which the length information is stored. On calling this function this parameter shall contain the size of the buffer provided by resultPtr. If the key element is configured |

| | | |
|---|---|---|
| | | to allow partial access, this parameter contains the amount of data which should be read from the key element. The size may not be equal to the size of the provided buffer anymore. When the request has finished, the amount of data that has been stored shall be stored. |
| **Parameters (out)** | `resultPtr` | Holds the pointer of the buffer for the returned key element. |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| | `CRYPTO_E_BUSY` | Request failed, Crypto Driver Object is busy |
| | `CRYPTO_E_KEY_NOT_AVAILABLE` | Request failed, the requested key element is not available |
| | `CRYPTO_E_KEY_READ_FAIL` | Request failed because read access was denied |
| | `CRYPTO_E_SMALL_BUFFER` | The provided buffer is too small to store the result |

#### 5.2.2.4.11. CryIf_KeyElementSet

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the set key element function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyElementSet`** `( uint32 cryIfKeyId , uint32 keyElementId , const uint8 * keyPtr , uint32 keyLength );` | |
| **Service ID** | CRYIF_SID_KEYELEMENTSET | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key whose key element shall be set. |
| | `keyElementId` | Holds the identifier of the key element which shall be set. |
| | `keyPtr` | Holds the pointer to the key data which shall be set as key element. |

| | | |
|---|---|---|
| | keyLength | Contains the length of the key element in bytes. |
| **Return Value** | Standard Return Value | |
| | E_OK | Request successful |
| | E_NOT_OK | Request failed |
| | CRYPTO_E_BUSY: | Request failed, Crypto Driver Object is busy |
| | CRYPTO_E_KEY_WRITE_FAIL: | Request failed because write access was denied |
| | CRYPTO_E_KEY_NOT_AVAILABLE: | Request failed because the key is not available. |
| | CRYPTO_E_KEY_SIZE_MISMATCH: | Request failed, key element size does not match size of provided data. |

#### 5.2.2.4.12. CryIf_KeyExchangeCalcPubVal

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the key exchange public value calculation function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyExchangeCalcPubVal`** `( uint32 cryIfKeyId , uint8 * publicValuePtr , uint32 * publicValueLengthPtr );` | |
| **Service ID** | CRYIF_SID_KEYEXCHANGECALCPUBVAL | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | cryIfKeyId | Holds the identifier of the key which shall be used for the key exchange protocol. |
| **Parameters (in,out)** | publicValueLengthPtr | Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | publicValuePtr | Contains the pointer to the data where the public value shall be stored. |
| **Return Value** | Standard Return Value | |
| | E_OK | Request successful |

| | | |
|---|---|---|
| | E_NOT_OK | Request failed |
| | E_BUSY | Request Failed, Crypto Driver Object is Busy |
| | CRYPTO_E_SMALL_BUFFER | The provided buffer is too small to store the result |

#### 5.2.2.4.13. CryIf_KeyExchangeCalcSecret

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the key exchange common shared secret calculation function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyExchangeCalcSecret`** `( uint32 cryIfKeyId , const uint8 * partnerPublicValuePtr , uint32 partnerPublicValueLength );` | |
| **Service ID** | CRYIF_SID_KEYEXCHANGECALCSECRET | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | cryIfKeyId | Holds the identifier of the key which shall be used for the key exchange protocol. |
| | partnerPublicValuePtr | Holds the pointer to the memory location which contains the partner's public value. |
| | partnerPublicValueLength | Contains the length of the partner's public value in bytes. |
| **Return Value** | Standard Return Value | |
| | E_OK | Request successful |
| | E_NOT_OK | Request failed |
| | E_BUSY | Request Failed, Crypto Driver Object is Busy |
| | CRYPTO_E_SMALL_BUFFER | The provided buffer is too small to store the result |

#### 5.2.2.4.14. CryIf_KeyGenerate

| | |
|---|---|
| **Purpose** | This function shall dispatch the key generate function to the configured crypto driver object. |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeyGenerate`** `( uint32 cryIfKeyId );` |
| **Service ID** | CRYIF_SID_KEYGENERATE |

| | | |
|---|---|---|
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key which is to be updated with the generated value. |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| | `CRYPTO_E_BUSY` | Request failed, Crypto Driver Object is busy |
| **Description** | {Sync or Async, depends on the configuration} | |

#### 5.2.2.4.15. CryIf_KeySetValid

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the set key valid function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_KeySetValid`** `( uint32 cryIfKeyId );` | |
| **Service ID** | CRYIF_SID_KEYSETVALID | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Identifier of the key that shall be set to valid |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| | `CRYPTO_E_BUSY` | Request Failed, Crypro Driver Object is Busy |

#### 5.2.2.4.16. CryIf_ProcessJob

| | | |
|---|---|---|
| **Purpose** | Processes a job received from the CSM. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_ProcessJob`** `( uint32 channelId , Crypto_JobType * job );` | |
| **Service ID** | CRYIF_SID_PROCESSJOB | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `channelId` | Holds the identifier of the Crypto Channel |

| | | |
|---|---|---|
| **Parameters (in,out)** | `job` | Holds a pointer to the job structure that shall be processed |
| **Return Value** | Standard Return Value extended by the Crypto Stack | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| | `CRYPTO_E_SMALL_BUFFER` | Provided buffer is too small to store the result |
| | `CRYPTO_E_QUEUE_FULL` | Queue within the crypto driver is full |
| **Description** | {Sync or Async, depends on the configuration} | |

#### 5.2.2.4.17. CryIf_RandomSeed

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the random seed function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`CryIf_RandomSeed`** `( uint32 cryIfKeyId , const uint8 * seedPtr , uint32 seedLength );` | |
| **Service ID** | CRYIF_SID_RANDOMSEED | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `cryIfKeyId` | Holds the identifier of the key for which a new seed shall be generated. |
| | `seedPtr` | Holds a pointer to the memory location which contains the data to feed the seed. |
| | `seedLength` | Contains the length of the seed in bytes. |
| **Return Value** | Standard Return Value | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request failed |
| **Description** | {Sync or Async, depends on the configuration} | |

## 5.2.3. Integration notes

### 5.2.3.1. Exclusive areas

Exclusive areas are not used by the `CryIf` module.

### 5.2.3.2. Production errors

Production errors are not reported by the `CryIf` module.

### 5.2.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

| Memory section |
|---|
| CODE |
| CONST_UNSPECIFIED |
| CONST_32 |
| VAR_INIT_BOOLEAN |

### 5.2.3.4. Integration requirements

**WARNING** 

**Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

#### 5.2.3.4.1. CryIf.Req.Integration_KeyMgmt

| | |
|---|---|
| **Description** | Key management functions are only available if at least one key exists in the configuration. Otherwise, they are disabled via compiler switch and thus cannot be called. This applies to the following functions:<br>- CryIf_KeyElementSet<br>- CryIf_KeySetValid<br>- CryIf_KeyElementGet<br>- CryIf_KeyElementCopy |

| | |
|---|---|
| | ▶ CryIf_KeyCopy<br>▶ CryIf_KeyGenerate<br>▶ CryIf_KeyDerive<br>▶ CryIf_KeyExchangeCalcPubVal<br>▶ CryIf_KeyExchangeCalcSecret<br>▶ CryIf_CertificateParse<br>▶ CryIf_CertificateVerify<br>▶ CryIf_RandomSeed |

#### 5.2.3.4.2. CryIf.Req.Integration_CryIfInit

| | |
|---|---|
| **Description** | CryIf_Init() shall be called during the start-up procedure of the ECU (by e.g. BswM) before any other API of the module is called. |

# 5.3. Csm

## 5.3.1. Configuration parameters

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CommonPublishedInformation | 1..1 | **Label:** Common Published Information<br>Common container, aggregated by all modules. It contains published information about vendor and versions. |
| CsmGeneral | 1..1 | **Label:** CsmGeneral<br>Container for common configuration options. |
| CsmCallbacks | 0..1 | **Label:** CsmCallbacks<br>Container for callback function configurations. |
| CsmJobs | 0..1 | **Label:** CsmJobs<br>Container for configuration of CSM jobs. |

| Containers included | | |
|---|---|---|
| CsmKeys | 0..1 | **Label:** CsmKeys<br><br>Container for CSM key configurations. |
| CsmPrimitives | 1..n | **Label:** CsmPrimitives<br><br>Container for configuration of CsmPrimitives. |
| CsmQueues | 0..1 | **Label:** CsmQueues<br><br>Container for CSM queue configurations. |
| CsmEbGeneral | 1..1 | Container for EB specific common configurations. |
| PublishedInformation | 1..1 | **Label:** EB Published Information<br>Additional published parameters not covered by Common-PublishedInformation container. |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| IMPLEMENTATION_CONFIG_VARIANT | 1..1 |

| Parameter Name | IMPLEMENTATION_CONFIG_VARIANT |
|---|---|
| **Label** | Config Variant |
| **Description** | Select the configuration variant. Currently only PreCompile is supported. |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | VariantPreCompile |
| **Range** | VariantPreCompile |

### 5.3.1.1. CommonPublishedInformation

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| ArMajorVersion | 1..1 |
| ArMinorVersion | 1..1 |
| ArPatchVersion | 1..1 |
| SwMajorVersion | 1..1 |

| Parameters included | |
|---|---|
| SwMinorVersion | 1..1 |
| SwPatchVersion | 1..1 |
| ModuleId | 1..1 |
| VendorId | 1..1 |
| Release | 1..1 |

| Parameter Name | ArMajorVersion |
|---|---|
| Label | AUTOSAR Major Version |
| Description | Major version number of AUTOSAR specification on which the appropriate implementation is based on. |
| Multiplicity | 1..1 |
| Type | INTEGER_LABEL |
| Default value | 4 |
| Configuration class | PublishedInformation: |
| Origin | Elektrobit Automotive GmbH |

| Parameter Name | ArMinorVersion |
|---|---|
| Label | AUTOSAR Minor Version |
| Description | Minor version number of AUTOSAR specification on which the appropriate implementation is based on. |
| Multiplicity | 1..1 |
| Type | INTEGER_LABEL |
| Default value | 3 |
| Configuration class | PublishedInformation: |
| Origin | Elektrobit Automotive GmbH |

| Parameter Name | ArPatchVersion |
|---|---|
| Label | AUTOSAR Patch Version |
| Description | Patch level version number of AUTOSAR specification on which the appropriate implementation is based on. |
| Multiplicity | 1..1 |
| Type | INTEGER_LABEL |
| Default value | 0 |

| | |
|---|---|
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwMajorVersion** |
|---|---|
| **Label** | Software Major Version |
| **Description** | Major version number of the vendor specific implementation of the module. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 3 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwMinorVersion** |
|---|---|
| **Label** | Software Minor Version |
| **Description** | Minor version number of the vendor specific implementation of the module. The numbering is vendor specific. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 1 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwPatchVersion** |
|---|---|
| **Label** | Software Patch Version |
| **Description** | Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 15 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **ModuleId** |
|---|---|

| | |
|---|---|
| **Label** | Numeric Module ID |
| **Description** | Module ID of this module from Module List |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 110 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **VendorId** |
|---|---|
| **Label** | Vendor ID |
| **Description** | Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 1 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **Release** |
|---|---|
| **Label** | Release Information |
| **Multiplicity** | 1..1 |
| **Type** | STRING_LABEL |
| **Default value** | |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

### 5.3.1.2. CsmGeneral

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmAsymPrivateKeyMaxLength | 0..1 |
| CsmAsymPublicKeyMaxLength | 0..1 |

| Parameters included | |
|---|---|
| CsmDevErrorDetect | 1..1 |
| CsmMainFunctionPeriod | 0..1 |
| CsmSymKeyMaxLength | 0..1 |
| CsmUseDeprecated | 1..1 |
| CsmVersionInfoApi | 1..1 |

| Parameter Name | CsmAsymPrivateKeyMaxLength | |
|---|---|---|
| Label | CsmAsymPrivateKeyMaxLength | |
| Description | Maximum length in bytes of an asymmetric public key for all algorithm.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| Multiplicity | 0..1 | |
| Type | INTEGER | |
| Default value | 1 | |
| Range | >=1<br><br><=4294967295 | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAsymPublicKeyMaxLength | |
|---|---|---|
| Label | CsmAsymPublicKeyMaxLength | |
| Description | Maximum length in bytes of an asymmetric key for all algorithm.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| Multiplicity | 0..1 | |
| Type | INTEGER | |
| Default value | 1 | |
| Range | >=1<br><br><=4294967295 | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |

| | | |
|---|---|---|
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDevErrorDetect** | |
|---|---|---|
| **Label** | CsmDevErrorDetect | |
| **Description** | Switches the development error detection and notification on or off.<br>► TRUE = detection and notification is enabled<br>► FALSE = detection and notification is disabled | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | true | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMainFunctionPeriod** | |
|---|---|---|
| **Label** | CsmMainFunctionPeriod | |
| **Description** | Specifies the period of main function Csm_MainFunction in seconds.<br>Range:<br>► Float : ]0 .. 4294967295] | |
| **Multiplicity** | 0..1 | |
| **Type** | FLOAT | |
| **Default value** | 0.01 | |
| **Range** | >0<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSymKeyMaxLength** |
|---|---|
| **Label** | CsmSymKeyMaxLength |
| **Description** | Maximum length in bytes of a symmetric key for all algorithm.<br>Range: |

| | |
|---|---|
| | ► Integer : 1 .. 4294967295 |
| **Multiplicity** | 0..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br>\<=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmUseDeprecated** |
|---|---|
| **Label** | CsmUseDeprecated |
| **Description** | Decides if the deprecated interfaces shall be used (Backwards combatibility). Currently this is not supported.<br>► TRUE = use deprecated interfaces<br>► FALSE = use normal interfaces |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |
| **Default value** | false |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmVersionInfoApi** |
|---|---|
| **Label** | CsmVersionInfoApi |
| **Description** | 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 |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |
| **Default value** | false |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.3. CsmCallbacks

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmCallback | 0..n | **Label:** CsmCallback<br><br>Container for configuration of a callback function. |

### 5.3.1.4. CsmCallback

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmCallbackFunc | 0..1 |
| CsmCallbackId | 1..1 |

| **Parameter Name** | **CsmCallbackFunc** | |
|---|---|---|
| **Label** | CsmCallbackFunc | |
| **Description** | Callback function to be called if an asynchronous operation has finished. The corresponding job has to be configured to be processed asynchronously.<br><br>► ENABLED = A C API callback whose name shall be specified will be used.<br>► DISABLED = A callback connected to the generated RTE RequiredPort for this callback will be used. | |
| **Multiplicity** | 0..1 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmCallbackId** |
|---|---|
| **Label** | CsmCallbackId |
| **Description** | Identifier of the callback function. It shall be consecutive, gapless and shall start from zero.<br><br>Range:<br><br>► Integer : 0 .. 4294967295 |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.5. CsmJobs

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmJob | 1..n | **Label:** CsmJob<br><br>Container for configuration of CSM job. The container name serves as a symbolic name for the identifier of a job configuration. |

### 5.3.1.6. CsmJob

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmJobId | 1..1 |
| CsmJobKeyRef | 1..1 |
| CsmJobPrimitiveCallbackRef | 0..1 |
| CsmJobPrimitiveCallbackUpdateNotification | 0..1 |
| CsmJobPrimitiveRef | 1..1 |
| CsmJobPriority | 1..1 |
| CsmJobQueueRef | 1..1 |
| CsmJobUsePort | 1..1 |

| | |
|---|---|
| **Parameter Name** | **CsmJobId** |
| **Label** | CsmJobId |
| **Description** | Identifier of the CSM job. It shall be consecutive, gapless and shall start from zero. |

| | |
|---|---|
| | Range:<br>► Integer : 0 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Range** | >=0<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmJobKeyRef** |
|---|---|
| **Label** | CsmJobKeyRef |
| **Description** | This parameter refers to the key which shall be used for the CsmPrimitive. It's possible to use a CsmKey for different jobs. |
| **Multiplicity** | 1..1 |
| **Type** | REFERENCE |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmJobPrimitiveCallbackRef** |
|---|---|
| **Label** | CsmJobPrimitiveCallbackRef |
| **Description** | This parameter refers to the used CsmCallback.<br>The referred CsmCallback is called when the crypto job has been finished. |
| **Multiplicity** | 0..1 |
| **Type** | REFERENCE |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmJobPrimitiveCallbackUpdateNotification** |
|---|---|
| **Label** | CsmJobPrimitiveCallbackUpdateNotification |
| **Description** | This parameter indicates, whether the callback function shall be called, if the UPDATE operation has been finished. |
| **Multiplicity** | 0..1 |

| | | |
|---|---|---|
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmJobPrimitiveRef** | |
|---|---|---|
| **Label** | CsmJobPrimitiveRef | |
| **Description** | This parameter refers to the used CsmPrimitive.<br><br>Different jobs may refer to one CsmPrimitive. The referred CsmPrimitive provides detailed information on the actual cryptographic routine. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmJobPriority** | |
|---|---|---|
| **Label** | CsmJobPriority | |
| **Description** | Priority of the job.<br><br>The higher the value, the higher the job's priority.<br><br>Range:<br><br>► Integer : 0 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 0 | |
| **Range** | >=0<br><br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmJobQueueRef** |
|---|---|
| **Label** | CsmJobQueueRef |
| **Description** | This parameter refers to the queue. |

| | |
|---|---|
| | The queue is used if the underlying crypto driver object is busy. The queue refers also to the channel which is used. |
| **Multiplicity** | 1..1 |
| **Type** | REFERENCE |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmJobUsePort** |
|---|---|
| **Label** | CsmJobUsePort |
| **Description** | Does the job need RTE interfaces?<br>► TRUE = the job needs RTE interfaces<br>► FALSE = the job needs no RTE interfaces |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |
| **Default value** | false |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.7. CsmKeys

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmKey | 0..n | **Label:** CsmKey<br>Container for configuration of a CSM key. The container name serves as a symbolic name for the identifier of a key configuration. |

### 5.3.1.8. CsmKey

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmKeyId | 1..1 |
| CsmKeyRef | 1..1 |

| Parameters included | |
|---|---|
| CsmKeyUsePort | 1..1 |

| Parameter Name | CsmKeyId | |
|---|---|---|
| Label | CsmKeyId | |
| Description | Identifier of the CsmKey. It shall be consecutive, gapless and shall start from zero.<br><br>Range:<br><br>► Integer : 0 .. 4294967295 | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=0<br>---<br><=4294967295 | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmKeyRef | |
|---|---|---|
| Label | CsmKeyRef | |
| Description | This parameter refers to the used CryIfKey. The underlying CryIfKey refers to a specific CryptoKey in the Crypto Driver. | |
| Multiplicity | 1..1 | |
| Type | SYMBOLIC-NAME-REFERENCE | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmKeyUsePort |
|---|---|
| Label | CsmKeyUsePort |
| Description | Does the key need RTE interfaces?<br><br>► TRUE = RTE interfaces used for this key<br>► FALSE = No RTE interfaces used for this key |
| Multiplicity | 1..1 |
| Type | BOOLEAN |
| Default value | false |

| | | |
|---|---|---|
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.9. CsmPrimitives

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmAEADDecrypt | 0..1 | **Label:** CsmAEADDecrypt<br>Configuration of AEAD decryption primitives. |
| CsmAEADEncrypt | 0..1 | **Label:** CsmAEADEncrypt<br>Configuration of AEAD encryption primitives. |
| CsmDecrypt | 0..1 | **Label:** CsmDecrypt<br>Configurations of Decryption primitives. |
| CsmEncrypt | 0..1 | **Label:** CsmEncrypt<br>Configurations of Encryption primitives. |
| CsmHash | 0..1 | **Label:** CsmHash<br>Container for Hash Configurations. |
| CsmMacGenerate | 0..1 | **Label:** CsmMacGenerate<br>Configurations of MacGenerate primitives. |
| CsmMacVerify | 0..1 | **Label:** CsmMacVerify<br>Configurations of MacVerify primitives. |
| CsmRandomGenerate | 0..1 | **Label:** CsmRandomGenerate<br>Configurations of RandomGenerate primitives. |
| CsmSecureCounter | 0..1 | **Label:** CsmSecureCounter<br>Configurations of SecureCounter primitives. |
| CsmSignatureGenerate | 0..1 | **Label:** CsmSignatureGenerate<br>Configurations of SignatureGenerate primitives. |
| CsmSignatureVerify | 0..1 | **Label:** CsmSignatureVerify<br>Configurations of SignatureVerify primitives. |

### 5.3.1.10. CsmAEADDecrypt

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmAEADDecryptConfig | 1..1 | **Label:** CsmAEADDecryptConfig<br><br>Container for configuration of a CSM decryption interface. The container name serves as a symbolic name for the identifier of an decryption interface. |

### 5.3.1.11. CsmAEADDecryptConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmAEADDecryptAlgorithmFamiliy | 1..1 |
| CsmAEADDecryptAlgorithmFamilyCustom | 0..1 |
| CsmAEADDecryptAlgorithmKeyLength | 1..1 |
| CsmAEADDecryptAlgorithmMode | 1..1 |
| CsmAEADDecryptAlgorithmModeCustom | 0..1 |
| CsmAEADDecryptAssociatedDataMaxLength | 1..1 |
| CsmAEADDecryptCiphertextMaxLength | 1..1 |
| CsmAEADDecryptKeyRef | 1..1 |
| CsmAEADDecryptPlaintextMaxLength | 1..1 |
| CsmAEADDecryptProcessing | 1..1 |
| CsmAEADDecryptQueueRef | 1..1 |
| CsmAEADDecryptTagLength | 1..1 |

| Parameter Name | CsmAEADDecryptAlgorithmFamiliy |
|---|---|
| **Label** | CsmAEADDecryptAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES |

| | |
|---|---|
| | ► CRYPTO_ALGOFAM_CUSTOM |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_AES |
| **Range** | CRYPTO_ALGOFAM_3DES<br>CRYPTO_ALGOFAM_AES<br>CRYPTO_ALGOFAM_CUSTOM |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmAEADDecryptAlgorithmFamilyCustom** |
|---|---|
| **Label** | CsmAEADDecryptAlgorithmFamilyCustom |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADDecryptAlgorithmFamiliy. |
| **Multiplicity** | 0..1 |
| **Type** | STRING |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmAEADDecryptAlgorithmKeyLength** |
|---|---|
| **Label** | CsmAEADDecryptAlgorithmKeyLength |
| **Description** | Size of the AEAD decryption key in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| Parameter Name | CsmAEADDecryptAlgorithmMode | |
|---|---|---|
| Label | CsmAEADDecryptAlgorithmMode | |
| Description | Determines the algorithm mode used for the crypto service.<br>Range:<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_GCM | |
| Multiplicity | 1..1 | |
| Type | ENUMERATION | |
| Default value | CRYPTO_ALGOMODE_GCM | |
| Range | CRYPTO_ALGOMODE_CUSTOM<br>CRYPTO_ALGOMODE_GCM | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAEADDecryptAlgorithmModeCustom | |
|---|---|---|
| Label | CsmAEADDecryptAlgorithmModeCustom | |
| Description | Name of the custom algorithm mode used for the crypto service. | |
| Multiplicity | 0..1 | |
| Type | STRING | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAEADDecryptAssociatedDataMaxLength |
|---|---|
| Label | CsmAEADDecryptAssociatedDataMaxLength |
| Description | Max size of the input associated data length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| Multiplicity | 1..1 |
| Type | INTEGER |
| Default value | 1 |
| Range | >=1 |

| | | |
|---|---|---|
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADDecryptCiphertextMaxLength** | |
|---|---|---|
| **Label** | CsmAEADDecryptCiphertextMaxLength | |
| **Description** | Max size of the input ciphertext in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br>---<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADDecryptKeyRef** | |
|---|---|---|
| **Label** | CsmAEADDecryptKeyRef | |
| **Description** | This parameter refers to the key used for that decryption primitive. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADDecryptPlaintextMaxLength** |
|---|---|
| **Label** | CsmAEADDecryptPlaintextMaxLength |
| **Description** | Size of the output plaintext length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |

| | | |
|---|---|---|
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADDecryptProcessing** |
|---|---|
| **Label** | CsmAEADDecryptProcessing |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_ASYNCHRONOUS |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmAEADDecryptQueueRef** |
|---|---|
| **Label** | CsmAEADDecryptQueueRef |
| **Description** | This parameter refers to the queue used for that decryption primitive. |
| **Multiplicity** | 1..1 |
| **Type** | REFERENCE |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmAEADDecryptTagLength** |
|---|---|
| **Label** | CsmAEADDecryptTagLength |

| | |
|---|---|
| **Description** | Size of the input Tag length in BITS.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.12. CsmAEADEncrypt

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmAEADEncryptConfig | 1..1 | **Label:** CsmAEADEncryptConfig<br><br>Container for configuration of a CSM encryption interface. The container name serves as a symbolic name for the identifier of an encryption interface. |

### 5.3.1.13. CsmAEADEncryptConfig

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmAEADEncryptAlgorithmFamiliy | 1..1 |
| CsmAEADEncryptAlgorithmFamilyCustom | 0..1 |
| CsmAEADEncryptAlgorithmKeyLength | 1..1 |
| CsmAEADEncryptAlgorithmMode | 1..1 |
| CsmAEADEncryptAlgorithmModeCustom | 0..1 |
| CsmAEADEncryptAssociatedDataMaxLength | 1..1 |
| CsmAEADEncryptCiphertextMaxLength | 1..1 |

| Parameters included | |
|---|---|
| CsmAEADEncryptKeyRef | 1..1 |
| CsmAEADEncryptPlaintextMaxLength | 1..1 |
| CsmAEADEncryptProcessing | 1..1 |
| CsmAEADEncryptQueueRef | 1..1 |
| CsmAEADEncryptTagLength | 1..1 |

| Parameter Name | CsmAEADEncryptAlgorithmFamiliy | |
|---|---|---|
| Label | CsmAEADEncryptAlgorithmFamiliy | |
| Description | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br>Range:<br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES<br>► CRYPTO_ALGOFAM_CUSTOM | |
| Multiplicity | 1..1 | |
| Type | ENUMERATION | |
| Default value | CRYPTO_ALGOFAM_AES | |
| Range | CRYPTO_ALGOFAM_3DES<br>CRYPTO_ALGOFAM_AES<br>CRYPTO_ALGOFAM_CUSTOM | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAEADEncryptAlgorithmFamilyCustom | |
|---|---|---|
| Label | CsmAEADEncryptAlgorithmFamilyCustom | |
| Description | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmAEADEncryptAlgorithmFamiliy. | |
| Multiplicity | 0..1 | |
| Type | STRING | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAEADEncryptAlgorithmKeyLength | |
|---|---|---|
| Label | CsmAEADEncryptAlgorithmKeyLength | |
| Description | Size of the AEAD encryption key in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Default value | 1 | |
| Range | >=1<br>---<br><=4294967295 | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAEADEncryptAlgorithmMode | |
|---|---|---|
| Label | CsmAEADEncryptAlgorithmMode | |
| Description | Determines the algorithm mode used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_GCM | |
| Multiplicity | 1..1 | |
| Type | ENUMERATION | |
| Default value | CRYPTO_ALGOMODE_GCM | |
| Range | CRYPTO_ALGOMODE_CUSTOM<br>---<br>CRYPTO_ALGOMODE_GCM | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmAEADEncryptAlgorithmModeCustom |
|---|---|
| Label | CsmAEADEncryptAlgorithmModeCustom |
| Description | Name of the custom algorithm mode used for the crypto service. |
| Multiplicity | 0..1 |
| Type | STRING |

| | | |
|---|---|---|
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptAssociatedDataMaxLength** | |
|---|---|---|
| **Label** | CsmAEADEncryptAssociatedDataMaxLength | |
| **Description** | Max size of the input associated data length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptCiphertextMaxLength** | |
|---|---|---|
| **Label** | CsmAEADEncryptCiphertextMaxLength | |
| **Description** | Max size of the output ciphertext length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptKeyRef** |
|---|---|
| **Label** | CsmAEADEncryptKeyRef |

| | | |
|---|---|---|
| **Description** | This parameter refers to the key used for that encryption primitive. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptPlaintextMaxLength** | |
|---|---|---|
| **Label** | CsmAEADEncryptPlaintextMaxLength | |
| **Description** | Max size of the input plaintext length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptProcessing** |
|---|---|
| **Label** | CsmAEADEncryptProcessing |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_ASYNCHRONOUS |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS |

| | | |
|---|---|---|
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptQueueRef** | |
|---|---|---|
| **Label** | CsmAEADEncryptQueueRef | |
| **Description** | This parameter refers to the queue used for that encryption primitive. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmAEADEncryptTagLength** | |
|---|---|---|
| **Label** | CsmAEADEncryptTagLength | |
| **Description** | Size of the output Tag length in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br>---<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.14. CsmDecrypt

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmDecryptConfig | 1..1 | **Label:** CsmDecryptConfig<br><br>Container for configuration of a CSM decryption interface. The container name serves as a symbolic name for the identifier of an decryption interface. |

### 5.3.1.15. CsmDecryptConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmDecryptAlgorithmFamiliy | 1..1 |
| CsmDecryptAlgorithmFamilyCustom | 0..1 |
| CsmDecryptAlgorithmKeyLength | 1..1 |
| CsmDecryptAlgorithmMode | 1..1 |
| CsmDecryptAlgorithmModeCustom | 0..1 |
| CsmDecryptAlgorithmSecondaryFamily | 1..1 |
| CsmDecryptAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmDecryptDataMaxLength | 1..1 |
| CsmDecryptProcessing | 1..1 |
| CsmDecryptResultMaxLength | 1..1 |

| **Parameter Name** | **CsmDecryptAlgorithmFamiliy** |
|---|---|
| **Label** | CsmDecryptAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES<br>► CRYPTO_ALGOFAM_CHACHA<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_ECIES<br>► CRYPTO_ALGOFAM_RSA |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_AES |
| **Range** | CRYPTO_ALGOFAM_3DES<br>CRYPTO_ALGOFAM_AES<br>CRYPTO_ALGOFAM_CHACHA<br>CRYPTO_ALGOFAM_CUSTOM |

| | | |
|---|---|---|
| | CRYPTO_ALGOFAM_ECIES | |
| | CRYPTO_ALGOFAM_RSA | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDecryptAlgorithmFamilyCustom** | |
|---|---|---|
| **Label** | CsmDecryptAlgorithmFamilyCustom | |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmDecryptAlgorithmFamiliy. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDecryptAlgorithmKeyLength** | |
|---|---|---|
| **Label** | CsmDecryptAlgorithmKeyLength | |
| **Description** | Size of the encryption key in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1 | |
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDecryptAlgorithmMode** |
|---|---|
| **Label** | CsmDecryptAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOMODE_12ROUNDS |

| | |
|---|---|
| | ▶ CRYPTO_ALGOMODE_20ROUNDS<br>▶ CRYPTO_ALGOMODE_8ROUNDS<br>▶ CRYPTO_ALGOMODE_CBC<br>▶ CRYPTO_ALGOMODE_CFB<br>▶ CRYPTO_ALGOMODE_CTR<br>▶ CRYPTO_ALGOMODE_CUSTOM<br>▶ CRYPTO_ALGOMODE_ECB<br>▶ CRYPTO_ALGOMODE_OFB<br>▶ CRYPTO_ALGOMODE_RSAES_OAEP<br>▶ CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5<br>▶ CRYPTO_ALGOMODE_XTS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_ECB |
| **Range** | CRYPTO_ALGOMODE_12ROUNDS<br>CRYPTO_ALGOMODE_20ROUNDS<br>CRYPTO_ALGOMODE_8ROUNDS<br>CRYPTO_ALGOMODE_CBC<br>CRYPTO_ALGOMODE_CFB<br>CRYPTO_ALGOMODE_CTR<br>CRYPTO_ALGOMODE_CUSTOM<br>CRYPTO_ALGOMODE_ECB<br>CRYPTO_ALGOMODE_OFB<br>CRYPTO_ALGOMODE_RSAES_OAEP<br>CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5<br>CRYPTO_ALGOMODE_XTS |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmDecryptAlgorithmModeCustom** |
|---|---|
| **Label** | CsmDecryptAlgorithmModeCustom |
| **Description** | Name of the custom algorithm mode used for the crypto service. |
| **Multiplicity** | 0..1 |

| | | |
|---|---|---|
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDecryptAlgorithmSecondaryFamily** | |
|---|---|---|
| **Label** | CsmDecryptAlgorithmSecondaryFamily | |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br>Range:<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CRYPTO_ALGOFAM_NOT_SET | |
| **Range** | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_NOT_SET | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDecryptAlgorithmSecondaryFamilyCustom** | |
|---|---|---|
| **Label** | CsmDecryptAlgorithmSecondaryFamilyCustom | |
| **Description** | Name of the custom secondary algorithm family used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmDecryptDataMaxLength** |
|---|---|
| **Label** | CsmDecryptDataMaxLength |
| **Description** | Max size of the input ciphertext length in bytes.<br>Range: |

| | |
|---|---|
| | ► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmDecryptProcessing** |
|---|---|
| **Label** | CsmDecryptProcessing |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_ASYNCHRONOUS |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmDecryptResultMaxLength** |
|---|---|
| **Label** | CsmDecryptResultMaxLength |
| **Description** | Max size of the output plaintext length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |

| | | |
|---|---|---|
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.16. CsmEncrypt

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmEncryptConfig | 1..1 | **Label:** CsmEncryptConfig<br><br>Container for configuration of a CSM encryption interface. The container name serves as a symbolic name for the identifier of an encryption interface. |

### 5.3.1.17. CsmEncryptConfig

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmEncryptAlgorithmFamiliy | 1..1 |
| CsmEncryptAlgorithmFamilyCustom | 0..1 |
| CsmEncryptAlgorithmKeyLength | 1..1 |
| CsmEncryptAlgorithmMode | 1..1 |
| CsmEncryptAlgorithmModeCustom | 0..1 |
| CsmEncryptAlgorithmSecondaryFamily | 1..1 |
| CsmEncryptAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmEncryptDataMaxLength | 1..1 |
| CsmEncryptProcessing | 1..1 |
| CsmEncryptResultMaxLength | 1..1 |

| **Parameter Name** | **CsmEncryptAlgorithmFamiliy** |
|---|---|
| **Label** | CsmEncryptAlgorithmFamiliy |

| | |
|---|---|
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES<br>► CRYPTO_ALGOFAM_CHACHA<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_ECIES<br>► CRYPTO_ALGOFAM_RSA |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_AES |
| **Range** | CRYPTO_ALGOFAM_3DES<br>CRYPTO_ALGOFAM_AES<br>CRYPTO_ALGOFAM_CHACHA<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_ECIES<br>CRYPTO_ALGOFAM_RSA |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| | |
|---|---|
| **Parameter Name** | **CsmEncryptAlgorithmFamilyCustom** |
| **Label** | CsmEncryptAlgorithmFamilyCustom |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmEncryptAlgorithmFamiliy. |
| **Multiplicity** | 0..1 |
| **Type** | STRING |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| | |
|---|---|
| **Parameter Name** | **CsmEncryptAlgorithmKeyLength** |
| **Label** | CsmEncryptAlgorithmKeyLength |

| | |
|---|---|
| **Description** | Size of the encryption key in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit |

| **Parameter Name** | **CsmEncryptAlgorithmMode** |
|---|---|
| **Label** | CsmEncryptAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOMODE_12ROUNDS<br>► CRYPTO_ALGOMODE_20ROUNDS<br>► CRYPTO_ALGOMODE_8ROUNDS<br>► CRYPTO_ALGOMODE_CBC<br>► CRYPTO_ALGOMODE_CFB<br>► CRYPTO_ALGOMODE_CTR<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_ECB<br>► CRYPTO_ALGOMODE_NOT_SET<br>► CRYPTO_ALGOMODE_OFB<br>► CRYPTO_ALGOMODE_RSAES_OAEP<br>► CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5<br>► CRYPTO_ALGOMODE_XTS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_ECB |

| | | |
|---|---|---|
| **Range** | CRYPTO_ALGOMODE_12ROUNDS | |
| | CRYPTO_ALGOMODE_20ROUNDS | |
| | CRYPTO_ALGOMODE_8ROUNDS | |
| | CRYPTO_ALGOMODE_CBC | |
| | CRYPTO_ALGOMODE_CFB | |
| | CRYPTO_ALGOMODE_CTR | |
| | CRYPTO_ALGOMODE_CUSTOM | |
| | CRYPTO_ALGOMODE_ECB | |
| | CRYPTO_ALGOMODE_NOT_SET | |
| | CRYPTO_ALGOMODE_OFB | |
| | CRYPTO_ALGOMODE_RSAES_OAEP | |
| | CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5 | |
| | CRYPTO_ALGOMODE_XTS | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| | | |
|---|---|---|
| **Parameter Name** | **CsmEncryptAlgorithmModeCustom** | |
| **Label** | CsmEncryptAlgorithmModeCustom | |
| **Description** | Name of the custom algorithm mode used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| | |
|---|---|
| **Parameter Name** | **CsmEncryptAlgorithmSecondaryFamily** |
| **Label** | CsmEncryptAlgorithmSecondaryFamily |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br>Range:<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |

| | | |
|---|---|---|
| **Default value** | CRYPTO_ALGOFAM_NOT_SET | |
| **Range** | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_NOT_SET | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmEncryptAlgorithmSecondaryFamilyCustom** | |
|---|---|---|
| **Label** | CsmEncryptAlgorithmSecondaryFamilyCustom | |
| **Description** | Name of the custom secondary algorithm family used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmEncryptDataMaxLength** | |
|---|---|---|
| **Label** | CsmEncryptDataMaxLength | |
| **Description** | Max size of the input plaintext length in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1 | |
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmEncryptProcessing** |
|---|---|
| **Label** | CsmEncryptProcessing |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback. |

| | |
|---|---|
| | Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_ASYNCHRONOUS |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmEncryptResultMaxLength** |
|---|---|
| **Label** | CsmEncryptResultMaxLength |
| **Description** | Max size of the output cipher length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.18. CsmHash

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmHashConfig | 1..1 | **Label:** CsmHashConfig<br>Container for configuration of a CSM hash. The container name serves as a symbolic name for the identifier of a key configuration. |

### 5.3.1.19. CsmHashConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmHashAlgorithmFamiliy | 1..1 |
| CsmHashAlgorithmFamilyCustom | 0..1 |
| CsmHashAlgorithmMode | 1..1 |
| CsmHashAlgorithmModeCustom | 0..1 |
| CsmHashAlgorithmSecondaryFamily | 1..1 |
| CsmHashAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmHashDataMaxLength | 1..1 |
| CsmHashProcessing | 1..1 |
| CsmHashResultLength | 1..1 |

| **Parameter Name** | **CsmHashAlgorithmFamiliy** |
|---|---|
| **Label** | CsmHashAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_BLAKE_1_256<br>► CRYPTO_ALGOFAM_BLAKE_1_512<br>► CRYPTO_ALGOFAM_BLAKE_2s_256<br>► CRYPTO_ALGOFAM_BLAKE_2s_512<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_RIPEMD160<br>► CRYPTO_ALGOFAM_SHA1<br>► CRYPTO_ALGOFAM_SHA2_224<br>► CRYPTO_ALGOFAM_SHA2_256<br>► CRYPTO_ALGOFAM_SHA2_384<br>► CRYPTO_ALGOFAM_SHA2_512<br>► CRYPTO_ALGOFAM_SHA2_512_224<br>► CRYPTO_ALGOFAM_SHA2_512_256<br>► CRYPTO_ALGOFAM_SHA3_224 |

| | |
|---|---|
| | ► CRYPTO_ALGOFAM_SHA3_256<br>► CRYPTO_ALGOFAM_SHA3_384<br>► CRYPTO_ALGOFAM_SHA3_512<br>► CRYPTO_ALGOFAM_SHA3_SHAKE128<br>► CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_SHA2_256 |
| **Range** | CRYPTO_ALGOFAM_BLAKE_1_256<br>CRYPTO_ALGOFAM_BLAKE_1_512<br>CRYPTO_ALGOFAM_BLAKE_2s_256<br>CRYPTO_ALGOFAM_BLAKE_2s_512<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_RIPEMD160<br>CRYPTO_ALGOFAM_SHA1<br>CRYPTO_ALGOFAM_SHA2_224<br>CRYPTO_ALGOFAM_SHA2_256<br>CRYPTO_ALGOFAM_SHA2_384<br>CRYPTO_ALGOFAM_SHA2_512<br>CRYPTO_ALGOFAM_SHA2_512_224<br>CRYPTO_ALGOFAM_SHA2_512_256<br>CRYPTO_ALGOFAM_SHA3_224<br>CRYPTO_ALGOFAM_SHA3_256<br>CRYPTO_ALGOFAM_SHA3_384<br>CRYPTO_ALGOFAM_SHA3_512<br>CRYPTO_ALGOFAM_SHA3_SHAKE128<br>CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmHashAlgorithmFamilyCustom** |
|---|---|
| **Label** | CsmHashAlgorithmFamilyCustom |

| | | |
|---|---|---|
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmHashAlgorithmFamiliy. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmHashAlgorithmMode** |
|---|---|
| **Label** | CsmHashAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br>Range:<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_NOT_SET |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_NOT_SET |
| **Range** | CRYPTO_ALGOMODE_CUSTOM |
| | CRYPTO_ALGOMODE_NOT_SET |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmHashAlgorithmModeCustom** | |
|---|---|---|
| **Label** | CsmHashAlgorithmModeCustom | |
| **Description** | Name of the custom algorithm mode used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmHashAlgorithmSecondaryFamily** |
|---|---|

| | |
|---|---|
| **Label** | CsmHashAlgorithmSecondaryFamily |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br>Range:<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_NOT_SET |
| **Range** | CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_NOT_SET |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmHashAlgorithmSecondaryFamilyCustom** |
|---|---|
| **Label** | CsmHashAlgorithmSecondaryFamilyCustom |
| **Description** | This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmHashAlgorithmSecondaryFamily. |
| **Multiplicity** | 0..1 |
| **Type** | STRING |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmHashDataMaxLength** |
|---|---|
| **Label** | CsmHashDataMaxLength |
| **Description** | Max size of the input data length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1 |

| | | |
|---|---|---|
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmHashProcessing** |
|---|---|
| **Label** | CsmHashProcessing |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_ASYNCHRONOUS |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmHashResultLength** |
|---|---|
| **Label** | CsmHashResultLength |
| **Description** | Size of the output hash length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.20. CsmMacGenerate

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmMacGenerateConfig | 1..1 | **Label:** CsmMacGenerateConfig<br><br>Container for configuration of a CSM mac generation interface. The container name serves as a symbolic name for the identifier of a MAC generation interface. |

### 5.3.1.21. CsmMacGenerateConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmMacGenerateAlgorithmFamiliy | 1..1 |
| CsmMacGenerateAlgorithmFamilyCustom | 0..1 |
| CsmMacGenerateAlgorithmKeyLength | 1..1 |
| CsmMacGenerateAlgorithmMode | 1..1 |
| CsmMacGenerateAlgorithmModeCustom | 0..1 |
| CsmMacGenerateAlgorithmSecondaryFamily | 1..1 |
| CsmMacGenerateAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmMacGenerateDataMaxLength | 1..1 |
| CsmMacGenerateProcessing | 1..1 |
| CsmMacGenerateResultLength | 1..1 |

| Parameter Name | CsmMacGenerateAlgorithmFamiliy |
|---|---|
| **Label** | CsmMacGenerateAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES<br>► CRYPTO_ALGOFAM_BLAKE_1_256<br>► CRYPTO_ALGOFAM_BLAKE_1_512<br>► CRYPTO_ALGOFAM_BLAKE_2s_256 |

| | |
|---|---|
| | ▶ CRYPTO_ALGOFAM_BLAKE_2s_512<br>▶ CRYPTO_ALGOFAM_CHACHA<br>▶ CRYPTO_ALGOFAM_CUSTOM<br>▶ CRYPTO_ALGOFAM_RIPEMD160<br>▶ CRYPTO_ALGOFAM_RNG<br>▶ CRYPTO_ALGOFAM_SHA1<br>▶ CRYPTO_ALGOFAM_SHA2_224<br>▶ CRYPTO_ALGOFAM_SHA2_256<br>▶ CRYPTO_ALGOFAM_SHA2_384<br>▶ CRYPTO_ALGOFAM_SHA2_512<br>▶ CRYPTO_ALGOFAM_SHA2_512_224<br>▶ CRYPTO_ALGOFAM_SHA2_512_256<br>▶ CRYPTO_ALGOFAM_SHA3_224<br>▶ CRYPTO_ALGOFAM_SHA3_256<br>▶ CRYPTO_ALGOFAM_SHA3_384<br>▶ CRYPTO_ALGOFAM_SHA3_512<br>▶ CRYPTO_ALGOFAM_SHA3_SHAKE128<br>▶ CRYPTO_ALGOFAM_SHA3_SHAKE256<br>▶ CRYPTO_ALGOFAM_SIPHASH |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_AES |
| **Range** | CRYPTO_ALGOFAM_3DES<br>CRYPTO_ALGOFAM_AES<br>CRYPTO_ALGOFAM_BLAKE_1_256<br>CRYPTO_ALGOFAM_BLAKE_1_512<br>CRYPTO_ALGOFAM_BLAKE_2s_256<br>CRYPTO_ALGOFAM_BLAKE_2s_512<br>CRYPTO_ALGOFAM_CHACHA<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_RIPEMD160<br>CRYPTO_ALGOFAM_RNG |

| | | |
|---|---|---|
| | CRYPTO_ALGOFAM_SHA1 | |
| | CRYPTO_ALGOFAM_SHA2_224 | |
| | CRYPTO_ALGOFAM_SHA2_256 | |
| | CRYPTO_ALGOFAM_SHA2_384 | |
| | CRYPTO_ALGOFAM_SHA2_512 | |
| | CRYPTO_ALGOFAM_SHA2_512_224 | |
| | CRYPTO_ALGOFAM_SHA2_512_256 | |
| | CRYPTO_ALGOFAM_SHA3_224 | |
| | CRYPTO_ALGOFAM_SHA3_256 | |
| | CRYPTO_ALGOFAM_SHA3_384 | |
| | CRYPTO_ALGOFAM_SHA3_512 | |
| | CRYPTO_ALGOFAM_SHA3_SHAKE128 | |
| | CRYPTO_ALGOFAM_SHA3_SHAKE256 | |
| | CRYPTO_ALGOFAM_SIPHASH | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacGenerateAlgorithmFamilyCustom** | |
|---|---|---|
| **Label** | CsmMacGenerateAlgorithmFamilyCustom | |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmMacGenerateAlgorithmFamiliy. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacGenerateAlgorithmKeyLength** |
|---|---|
| **Label** | CsmMacGenerateAlgorithmKeyLength |
| **Description** | Size of the MAC key in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |

| | | |
|---|---|---|
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| | | |
|---|---|---|
| **Parameter Name** | **CsmMacGenerateAlgorithmMode** | |
| **Label** | CsmMacGenerateAlgorithmMode | |
| **Description** | Determines the algorithm mode used for the crypto service.<br>Range:<br>► CRYPTO_ALGOMODE_CMAC<br>► CRYPTO_ALGOMODE_CTRDRBG<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_GMAC<br>► CRYPTO_ALGOMODE_HMAC<br>► CRYPTO_ALGOMODE_NOT_SET<br>► CRYPTO_ALGOMODE_SIPHASH_2_4<br>► CRYPTO_ALGOMODE_SIPHASH_4_8 | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CRYPTO_ALGOMODE_NOT_SET | |
| **Range** | CRYPTO_ALGOMODE_CMAC<br>CRYPTO_ALGOMODE_CTRDRBG<br>CRYPTO_ALGOMODE_CUSTOM<br>CRYPTO_ALGOMODE_GMAC<br>CRYPTO_ALGOMODE_HMAC<br>CRYPTO_ALGOMODE_NOT_SET<br>CRYPTO_ALGOMODE_SIPHASH_2_4<br>CRYPTO_ALGOMODE_SIPHASH_4_8 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| Parameter Name | CsmMacGenerateAlgorithmModeCustom | |
|---|---|---|
| Label | CsmMacGenerateAlgorithmModeCustom | |
| Description | Name of the custom algorithm mode used for the crypto service. | |
| Multiplicity | 0..1 | |
| Type | STRING | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmMacGenerateAlgorithmSecondaryFamily | |
|---|---|---|
| Label | CsmMacGenerateAlgorithmSecondaryFamily | |
| Description | Determines the secondary algorithm family used for the crypto service.<br>Range:<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET | |
| Multiplicity | 1..1 | |
| Type | ENUMERATION | |
| Default value | CRYPTO_ALGOFAM_NOT_SET | |
| Range | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_NOT_SET | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmMacGenerateAlgorithmSecondaryFamilyCustom | |
|---|---|---|
| Label | CsmMacGenerateAlgorithmSecondaryFamilyCustom | |
| Description | This is the second name of the custom algorithm family, if CRYPTO_ALGO-FAM_CUSTOM is set as CsmMacGenerateAlgorithmSecondaryFamily. | |
| Multiplicity | 0..1 | |
| Type | STRING | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmMacGenerateDataMaxLength | |
|---|---|---|
| Label | CsmMacGenerateDataMaxLength | |
| Description | Max size of the input data length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Default value | 1 | |
| Range | >=1<br><=4294967295 | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmMacGenerateProcessing | |
|---|---|---|
| Label | CsmMacGenerateProcessing | |
| Description | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS | |
| Multiplicity | 1..1 | |
| Type | ENUMERATION | |
| Default value | CSM_ASYNCHRONOUS | |
| Range | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS | |
| Configuration class | **VariantPreCompile:** | VariantPreCompile |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | CsmMacGenerateResultLength |
|---|---|
| Label | CsmMacGenerateResultLength |
| Description | Size of the output MAC length in bytes. |

| | |
|---|---|
| | Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.22. CsmMacVerify

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmMacVerifyConfig | 1..1 | **Label:** CsmMacVerifyConfig<br>Container for configuration of a CSM MAC verification interface. The container name serves as a symbolic name for the identifier of a MAC generation interface. |

### 5.3.1.23. CsmMacVerifyConfig

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmMacVerifyAlgorithmFamiliy | 1..1 |
| CsmMacVerifyAlgorithmFamilyCustom | 0..1 |
| CsmMacVerifyAlgorithmKeyLength | 1..1 |
| CsmMacVerifyAlgorithmMode | 1..1 |
| CsmMacVerifyAlgorithmModeCustom | 0..1 |
| CsmMacVerifyAlgorithmSecondaryFamily | 1..1 |
| CsmMacVerifyAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmMacVerifyCompareLength | 1..1 |
| CsmMacVerifyDataMaxLength | 1..1 |

| Parameters included | |
|---|---|
| CsmMacVerifyProcessing | 1..1 |

| Parameter Name | CsmMacVerifyAlgorithmFamiliy |
|---|---|
| Label | CsmMacVerifyAlgorithmFamiliy |
| Description | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES<br>► CRYPTO_ALGOFAM_BLAKE_1_256<br>► CRYPTO_ALGOFAM_BLAKE_1_512<br>► CRYPTO_ALGOFAM_BLAKE_2s_256<br>► CRYPTO_ALGOFAM_BLAKE_2s_512<br>► CRYPTO_ALGOFAM_CHACHA<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_RIPEMD160<br>► CRYPTO_ALGOFAM_RNG<br>► CRYPTO_ALGOFAM_SHA1<br>► CRYPTO_ALGOFAM_SHA2_224<br>► CRYPTO_ALGOFAM_SHA2_256<br>► CRYPTO_ALGOFAM_SHA2_384<br>► CRYPTO_ALGOFAM_SHA2_512<br>► CRYPTO_ALGOFAM_SHA2_512_224<br>► CRYPTO_ALGOFAM_SHA2_512_256<br>► CRYPTO_ALGOFAM_SHA3_224<br>► CRYPTO_ALGOFAM_SHA3_256<br>► CRYPTO_ALGOFAM_SHA3_384<br>► CRYPTO_ALGOFAM_SHA3_512<br>► CRYPTO_ALGOFAM_SHA3_SHAKE128<br>► CRYPTO_ALGOFAM_SHA3_SHAKE256<br>► CRYPTO_ALGOFAM_SIPHASH |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CRYPTO_ALGOFAM_AES | |
| **Range** | CRYPTO_ALGOFAM_3DES | |
| | CRYPTO_ALGOFAM_AES | |
| | CRYPTO_ALGOFAM_BLAKE_1_256 | |
| | CRYPTO_ALGOFAM_BLAKE_1_512 | |
| | CRYPTO_ALGOFAM_BLAKE_2s_256 | |
| | CRYPTO_ALGOFAM_BLAKE_2s_512 | |
| | CRYPTO_ALGOFAM_CHACHA | |
| | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_RIPEMD160 | |
| | CRYPTO_ALGOFAM_RNG | |
| | CRYPTO_ALGOFAM_SHA1 | |
| | CRYPTO_ALGOFAM_SHA2_224 | |
| | CRYPTO_ALGOFAM_SHA2_256 | |
| | CRYPTO_ALGOFAM_SHA2_384 | |
| | CRYPTO_ALGOFAM_SHA2_512 | |
| | CRYPTO_ALGOFAM_SHA2_512_224 | |
| | CRYPTO_ALGOFAM_SHA2_512_256 | |
| | CRYPTO_ALGOFAM_SHA3_224 | |
| | CRYPTO_ALGOFAM_SHA3_256 | |
| | CRYPTO_ALGOFAM_SHA3_384 | |
| | CRYPTO_ALGOFAM_SHA3_512 | |
| | CRYPTO_ALGOFAM_SHA3_SHAKE128 | |
| | CRYPTO_ALGOFAM_SHA3_SHAKE256 | |
| | CRYPTO_ALGOFAM_SIPHASH | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacVerifyAlgorithmFamilyCustom** |
|---|---|
| **Label** | CsmMacVerifyAlgorithmFamilyCustom |

| | | |
|---|---|---|
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmMacVerifyAlgorithmFamiliy. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacVerifyAlgorithmKeyLength** | |
|---|---|---|
| **Label** | CsmMacVerifyAlgorithmKeyLength | |
| **Description** | Size of the MAC key in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1 | |
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | Elektrobit | |

| **Parameter Name** | **CsmMacVerifyAlgorithmMode** |
|---|---|
| **Label** | CsmMacVerifyAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br>Range:<br>► CRYPTO_ALGOMODE_CMAC<br>► CRYPTO_ALGOMODE_CTRDRBG<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_GMAC<br>► CRYPTO_ALGOMODE_HMAC<br>► CRYPTO_ALGOMODE_NOT_SET<br>► CRYPTO_ALGOMODE_SIPHASH_2_4 |

| | |
|---|---|
| | ► CRYPTO_ALGOMODE_SIPHASH_4_8 |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_NOT_SET |
| **Range** | CRYPTO_ALGOMODE_CMAC<br>CRYPTO_ALGOMODE_CTRDRBG<br>CRYPTO_ALGOMODE_CUSTOM<br>CRYPTO_ALGOMODE_GMAC<br>CRYPTO_ALGOMODE_HMAC<br>CRYPTO_ALGOMODE_NOT_SET<br>CRYPTO_ALGOMODE_SIPHASH_2_4<br>CRYPTO_ALGOMODE_SIPHASH_4_8 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit |

| **Parameter Name** | **CsmMacVerifyAlgorithmModeCustom** |
|---|---|
| **Label** | CsmMacVerifyAlgorithmModeCustom |
| **Description** | Name of the custom algorithm mode used for the crypto service. |
| **Multiplicity** | 0..1 |
| **Type** | STRING |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit |

| **Parameter Name** | **CsmMacVerifyAlgorithmSecondaryFamily** |
|---|---|
| **Label** | CsmMacVerifyAlgorithmSecondaryFamily |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_NOT_SET |

| | | |
|---|---|---|
| **Range** | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_NOT_SET | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacVerifyAlgorithmSecondaryFamilyCustom** | |
|---|---|---|
| **Label** | CsmMacVerifyAlgorithmSecondaryFamilyCustom | |
| **Description** | This is the second the name of the custom algorithm, if CRYPTO_ALGO-FAM_CUSTOM is set as CsmMacVerifyAlgorithmSecondaryFamily. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacVerifyCompareLength** | |
|---|---|---|
| **Label** | CsmMacVerifyCompareLength | |
| **Description** | Size of the input MAC length, that shall be verified, in BITS.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1 | |
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacVerifyDataMaxLength** |
|---|---|
| **Label** | CsmMacVerifyDataMaxLength |
| **Description** | Max size of the input data length, for whichs MAC shall be verified, in bytes.<br><br>Range:<br><br>► Integer : 1 .. 4294967295 |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmMacVerifyProcessing** |
|---|---|
| **Label** | CsmMacVerifyProcessing |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br><br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_ASYNCHRONOUS |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.24. CsmRandomGenerate

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmRandomGenerateConfig | 1..1 | **Label:** CsmRandomGenerateConfig<br><br>Container for configuration of a CSM random generator. The container name serves as a symbolic name for the identifier of a random generator configuration. |

### 5.3.1.25. CsmRandomGenerateConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmRandomGenerateAlgorithmFamiliy | 1..1 |
| CsmRandomGenerateAlgorithmFamilyCustom | 0..1 |
| CsmRandomGenerateAlgorithmMode | 1..1 |
| CsmRandomGenerateAlgorithmModeCustom | 0..1 |
| CsmRandomGenerateAlgorithmSecondaryFamily | 1..1 |
| CsmRandomGenerateAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmRandomGenerateProcessing | 1..1 |
| CsmRandomGenerateResultLength | 1..1 |

| **Parameter Name** | **CsmRandomGenerateAlgorithmFamiliy** |
|---|---|
| **Label** | CsmRandomGenerateAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_3DES<br>► CRYPTO_ALGOFAM_AES<br>► CRYPTO_ALGOFAM_BLAKE_1_256<br>► CRYPTO_ALGOFAM_BLAKE_1_512<br>► CRYPTO_ALGOFAM_BLAKE_2s_256<br>► CRYPTO_ALGOFAM_BLAKE_2s_512<br>► CRYPTO_ALGOFAM_CHACHA<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_RIPEMD160<br>► CRYPTO_ALGOFAM_RNG<br>► CRYPTO_ALGOFAM_SHA1<br>► CRYPTO_ALGOFAM_SHA2_224<br>► CRYPTO_ALGOFAM_SHA2_256<br>► CRYPTO_ALGOFAM_SHA2_384<br>► CRYPTO_ALGOFAM_SHA2_512 |

| | |
|---|---|
| | ▶ CRYPTO_ALGOFAM_SHA2_512_224<br>▶ CRYPTO_ALGOFAM_SHA2_512_256<br>▶ CRYPTO_ALGOFAM_SHA3_224<br>▶ CRYPTO_ALGOFAM_SHA3_256<br>▶ CRYPTO_ALGOFAM_SHA3_384<br>▶ CRYPTO_ALGOFAM_SHA3_512<br>▶ CRYPTO_ALGOFAM_SHA3_SHAKE128<br>▶ CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_AES |
| **Range** | CRYPTO_ALGOFAM_3DES<br>CRYPTO_ALGOFAM_AES<br>CRYPTO_ALGOFAM_BLAKE_1_256<br>CRYPTO_ALGOFAM_BLAKE_1_512<br>CRYPTO_ALGOFAM_BLAKE_2s_256<br>CRYPTO_ALGOFAM_BLAKE_2s_512<br>CRYPTO_ALGOFAM_CHACHA<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_RIPEMD160<br>CRYPTO_ALGOFAM_RNG<br>CRYPTO_ALGOFAM_SHA1<br>CRYPTO_ALGOFAM_SHA2_224<br>CRYPTO_ALGOFAM_SHA2_256<br>CRYPTO_ALGOFAM_SHA2_384<br>CRYPTO_ALGOFAM_SHA2_512<br>CRYPTO_ALGOFAM_SHA2_512_224<br>CRYPTO_ALGOFAM_SHA2_512_256<br>CRYPTO_ALGOFAM_SHA3_224<br>CRYPTO_ALGOFAM_SHA3_256<br>CRYPTO_ALGOFAM_SHA3_384<br>CRYPTO_ALGOFAM_SHA3_512 |

| | | |
|---|---|---|
| | CRYPTO_ALGOFAM_SHA3_SHAKE128 | |
| | CRYPTO_ALGOFAM_SHA3_SHAKE256 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateAlgorithmFamilyCustom** | |
|---|---|---|
| **Label** | CsmRandomGenerateAlgorithmFamilyCustom | |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmRandomAlgorithmFamiliy. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateAlgorithmMode** |
|---|---|
| **Label** | CsmRandomGenerateAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOMODE_CMAC<br>► CRYPTO_ALGOMODE_CTRDRBG<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_GMAC<br>► CRYPTO_ALGOMODE_HMAC<br>► CRYPTO_ALGOMODE_NOT_SET<br>► CRYPTO_ALGOMODE_SIPHASH_2_4<br>► CRYPTO_ALGOMODE_SIPHASH_4_8 |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_NOT_SET |
| **Range** | CRYPTO_ALGOMODE_CMAC |
| | CRYPTO_ALGOMODE_CTRDRBG |
| | CRYPTO_ALGOMODE_CUSTOM |

| | | |
|---|---|---|
| | CRYPTO_ALGOMODE_GMAC | |
| | CRYPTO_ALGOMODE_HMAC | |
| | CRYPTO_ALGOMODE_NOT_SET | |
| | CRYPTO_ALGOMODE_SIPHASH_2_4 | |
| | CRYPTO_ALGOMODE_SIPHASH_4_8 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateAlgorithmModeCustom** | |
|---|---|---|
| **Label** | CsmRandomGenerateAlgorithmModeCustom | |
| **Description** | Name of the custom algorithm mode used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateAlgorithmSecondaryFamily** | |
|---|---|---|
| **Label** | CsmRandomGenerateAlgorithmSecondaryFamily | |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br>Range:<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CRYPTO_ALGOFAM_NOT_SET | |
| **Range** | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_NOT_SET | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateAlgorithmSecondaryFamilyCustom** |
|---|---|
| **Label** | CsmRandomGenerateAlgorithmSecondaryFamilyCustom |

| | | |
|---|---|---|
| **Description** | Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is set as CsmRandomAlgorithmSecondaryFamily. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateProcessing** | |
|---|---|---|
| **Label** | CsmRandomGenerateProcessing | |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CSM_ASYNCHRONOUS | |
| **Range** | CSM_ASYNCHRONOUS | |
| | CSM_SYNCHRONOUS | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmRandomGenerateResultLength** |
|---|---|
| **Label** | CsmRandomGenerateResultLength |
| **Description** | Size of the random generate key in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |

| | | |
|---|---|---|
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.26. CsmSecureCounter

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmSecureCounterConfig | 1..1 | **Label:** CsmSecureCounterConfig<br><br>Container for configuration of a CSM counter. The container name serves as a symbolic name for the identifier of a secure counter configuration. |

### 5.3.1.27. CsmSecureCounterConfig

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmSecureCounterQueueRef | 1..1 |

| **Parameter Name** | **CsmSecureCounterQueueRef** | |
|---|---|---|
| **Label** | CsmSecureCounterQueueRef | |
| **Description** | This parameter refers to the queue used for that secure counter. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.28. CsmSignatureGenerate

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |

| Containers included | | |
|---|---|---|
| CsmSignatureGenerateConfig | 1..1 | **Label:** CsmSignatureGenerateConfig<br><br>Container for configuration of a CSM signature generation interface. The container name serves as a symbolic name for the identifier of signature generation interface. |

### 5.3.1.29. CsmSignatureGenerateConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmSignatureGenerateAlgorithmFamiliy | 1..1 |
| CsmSignatureGenerateAlgorithmFamilyCustom | 0..1 |
| CsmSignatureGenerateAlgorithmMode | 1..1 |
| CsmSignatureGenerateAlgorithmModeCustom | 0..1 |
| CsmSignatureGenerateAlgorithmSecondaryFamily | 1..1 |
| CsmSignatureGenerateAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmSignatureGenerateDataMaxLength | 1..1 |
| CsmSignatureGenerateKeyLength | 1..1 |
| CsmSignatureGenerateProcessing | 1..1 |
| CsmSignatureGenerateResultLength | 1..1 |

| Parameter Name | CsmSignatureGenerateAlgorithmFamiliy |
|---|---|
| **Label** | CsmSignatureGenerateAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_BRAINPOOL<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_ECCNIST<br>► CRYPTO_ALGOFAM_ED25519<br>► CRYPTO_ALGOFAM_RSA |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |

| | | |
|---|---|---|
| **Default value** | CRYPTO_ALGOFAM_BRAINPOOL | |
| **Range** | CRYPTO_ALGOFAM_BRAINPOOL | |
| | CRYPTO_ALGOFAM_CUSTOM | |
| | CRYPTO_ALGOFAM_ECCNIST | |
| | CRYPTO_ALGOFAM_ED25519 | |
| | CRYPTO_ALGOFAM_RSA | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateAlgorithmFamilyCustom** | |
|---|---|---|
| **Label** | CsmSignatureGenerateAlgorithmFamilyCustom | |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureGenerateAlgorithmFamiliy. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateAlgorithmMode** |
|---|---|
| **Label** | CsmSignatureGenerateAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_NOT_SET<br>► CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5<br>► CRYPTO_ALGOMODE_RSASSA_PSS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_NOT_SET |
| **Range** | CRYPTO_ALGOMODE_CUSTOM |
| | CRYPTO_ALGOMODE_NOT_SET |

| | | |
|---|---|---|
| | CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5 | |
| | CRYPTO_ALGOMODE_RSASSA_PSS | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateAlgorithmModeCustom** | |
|---|---|---|
| **Label** | CsmSignatureGenerateAlgorithmModeCustom | |
| **Description** | Name of the custom algorithm mode used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateAlgorithmSecondaryFamily** |
|---|---|
| **Label** | CsmSignatureGenerateAlgorithmSecondaryFamily |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_BLAKE_1_256<br>► CRYPTO_ALGOFAM_BLAKE_1_512<br>► CRYPTO_ALGOFAM_BLAKE_2s_256<br>► CRYPTO_ALGOFAM_BLAKE_2s_512<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET<br>► CRYPTO_ALGOFAM_RIPEMD160<br>► CRYPTO_ALGOFAM_SHA1<br>► CRYPTO_ALGOFAM_SHA2_224<br>► CRYPTO_ALGOFAM_SHA2_256<br>► CRYPTO_ALGOFAM_SHA2_384<br>► CRYPTO_ALGOFAM_SHA2_512<br>► CRYPTO_ALGOFAM_SHA2_512_224<br>► CRYPTO_ALGOFAM_SHA2_512_256 |

| | |
|---|---|
| | ► CRYPTO_ALGOFAM_SHA3_224<br>► CRYPTO_ALGOFAM_SHA3_256<br>► CRYPTO_ALGOFAM_SHA3_384<br>► CRYPTO_ALGOFAM_SHA3_512<br>► CRYPTO_ALGOFAM_SHA3_SHAKE128<br>► CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_NOT_SET |
| **Range** | CRYPTO_ALGOFAM_BLAKE_1_256<br>CRYPTO_ALGOFAM_BLAKE_1_512<br>CRYPTO_ALGOFAM_BLAKE_2s_256<br>CRYPTO_ALGOFAM_BLAKE_2s_512<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_NOT_SET<br>CRYPTO_ALGOFAM_RIPEMD160<br>CRYPTO_ALGOFAM_SHA1<br>CRYPTO_ALGOFAM_SHA2_224<br>CRYPTO_ALGOFAM_SHA2_256<br>CRYPTO_ALGOFAM_SHA2_384<br>CRYPTO_ALGOFAM_SHA2_512<br>CRYPTO_ALGOFAM_SHA2_512_224<br>CRYPTO_ALGOFAM_SHA2_512_256<br>CRYPTO_ALGOFAM_SHA3_224<br>CRYPTO_ALGOFAM_SHA3_256<br>CRYPTO_ALGOFAM_SHA3_384<br>CRYPTO_ALGOFAM_SHA3_512<br>CRYPTO_ALGOFAM_SHA3_SHAKE128<br>CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmSignatureGenerateAlgorithmSecondaryFamilyCustom** |
|---|---|

| | | |
|---|---|---|
| **Label** | CsmSignatureGenerateAlgorithmSecondaryFamilyCustom | |
| **Description** | Name of the custom secondary algorithm family used for the crypto service. This is the second name of the custom algorithm family, if CRYPTO_ALGO-FAM_CUSTOM is set as CsmSignatureGenerateAlgorithmSecondaryFamily. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateDataMaxLength** | |
|---|---|---|
| **Label** | CsmSignatureGenerateDataMaxLength | |
| **Description** | Size of the input data length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1 | |
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateKeyLength** |
|---|---|
| **Label** | CsmSignatureGenerateKeyLength |
| **Description** | Size of the signature generate key in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1 |

| | | |
|---|---|---|
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateProcessing** | |
|---|---|---|
| **Label** | CsmSignatureGenerateProcessing | |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CSM_ASYNCHRONOUS | |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureGenerateResultLength** | |
|---|---|---|
| **Label** | CsmSignatureGenerateResultLength | |
| **Description** | Size of the output signature length in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.30. CsmSignatureVerify

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmSignatureVerifyConfig | 1..1 | **Label:** CsmSignatureVerifyConfig<br><br>Container for configuration of a CSM signature verification interface. The container name serves as a symbolic name for the identifier of signature verification interface. |

### 5.3.1.31. CsmSignatureVerifyConfig

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmSignatureVerifyAlgorithmFamiliy | 1..1 |
| CsmSignatureVerifyAlgorithmFamilyCustom | 0..1 |
| CsmSignatureVerifyAlgorithmMode | 1..1 |
| CsmSignatureVerifyAlgorithmModeCustom | 0..1 |
| CsmSignatureVerifyAlgorithmSecondaryFamily | 1..1 |
| CsmSignatureVerifyAlgorithmSecondaryFamilyCustom | 0..1 |
| CsmSignatureVerifyCompareLength | 1..1 |
| CsmSignatureVerifyDataMaxLength | 1..1 |
| CsmSignatureVerifyKeyLength | 1..1 |
| CsmSignatureVerifyProcessing | 1..1 |

| **Parameter Name** | **CsmSignatureVerifyAlgorithmFamiliy** |
|---|---|
| **Label** | CsmSignatureVerifyAlgorithmFamiliy |
| **Description** | Determines the algorithm family used for the crypto service. This parameter defines the most significant part of the algorithm.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_BRAINPOOL<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_ECCNIST<br>► CRYPTO_ALGOFAM_ED25519<br>► CRYPTO_ALGOFAM_RSA |

| | |
|---|---|
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_BRAINPOOL |
| **Range** | CRYPTO_ALGOFAM_BRAINPOOL<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_ECCNIST<br>CRYPTO_ALGOFAM_ED25519<br>CRYPTO_ALGOFAM_RSA |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmSignatureVerifyAlgorithmFamilyCustom** |
|---|---|
| **Label** | CsmSignatureVerifyAlgorithmFamilyCustom |
| **Description** | This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureVerifyAlgorithmFamiliy. |
| **Multiplicity** | 0..1 |
| **Type** | STRING |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile<br>**VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmSignatureVerifyAlgorithmMode** |
|---|---|
| **Label** | CsmSignatureVerifyAlgorithmMode |
| **Description** | Determines the algorithm mode used for the crypto service.<br>Range:<br>► CRYPTO_ALGOMODE_CUSTOM<br>► CRYPTO_ALGOMODE_NOT_SET<br>► CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5<br>► CRYPTO_ALGOMODE_RSASSA_PSS |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOMODE_NOT_SET |
| **Range** | CRYPTO_ALGOMODE_CUSTOM |

| | | |
|---|---|---|
| | CRYPTO_ALGOMODE_NOT_SET | |
| | CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5 | |
| | CRYPTO_ALGOMODE_RSASSA_PSS | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureVerifyAlgorithmModeCustom** | |
|---|---|---|
| **Label** | CsmSignatureVerifyAlgorithmModeCustom | |
| **Description** | Name of the custom algorithm mode used for the crypto service. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureVerifyAlgorithmSecondaryFamily** |
|---|---|
| **Label** | CsmSignatureVerifyAlgorithmSecondaryFamily |
| **Description** | Determines the secondary algorithm family used for the crypto service.<br><br>Range:<br><br>► CRYPTO_ALGOFAM_BLAKE_1_256<br>► CRYPTO_ALGOFAM_BLAKE_1_512<br>► CRYPTO_ALGOFAM_BLAKE_2s_256<br>► CRYPTO_ALGOFAM_BLAKE_2s_512<br>► CRYPTO_ALGOFAM_CUSTOM<br>► CRYPTO_ALGOFAM_NOT_SET<br>► CRYPTO_ALGOFAM_RIPEMD160<br>► CRYPTO_ALGOFAM_SHA1<br>► CRYPTO_ALGOFAM_SHA2_224<br>► CRYPTO_ALGOFAM_SHA2_256<br>► CRYPTO_ALGOFAM_SHA2_384<br>► CRYPTO_ALGOFAM_SHA2_512<br>► CRYPTO_ALGOFAM_SHA2_512_224<br>► CRYPTO_ALGOFAM_SHA2_512_256 |

| | |
|---|---|
| | ► CRYPTO_ALGOFAM_SHA3_224<br>► CRYPTO_ALGOFAM_SHA3_256<br>► CRYPTO_ALGOFAM_SHA3_384<br>► CRYPTO_ALGOFAM_SHA3_512<br>► CRYPTO_ALGOFAM_SHA3_SHAKE128<br>► CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CRYPTO_ALGOFAM_NOT_SET |
| **Range** | CRYPTO_ALGOFAM_BLAKE_1_256<br>CRYPTO_ALGOFAM_BLAKE_1_512<br>CRYPTO_ALGOFAM_BLAKE_2s_256<br>CRYPTO_ALGOFAM_BLAKE_2s_512<br>CRYPTO_ALGOFAM_CUSTOM<br>CRYPTO_ALGOFAM_NOT_SET<br>CRYPTO_ALGOFAM_RIPEMD160<br>CRYPTO_ALGOFAM_SHA1<br>CRYPTO_ALGOFAM_SHA2_224<br>CRYPTO_ALGOFAM_SHA2_256<br>CRYPTO_ALGOFAM_SHA2_384<br>CRYPTO_ALGOFAM_SHA2_512<br>CRYPTO_ALGOFAM_SHA2_512_224<br>CRYPTO_ALGOFAM_SHA2_512_256<br>CRYPTO_ALGOFAM_SHA3_224<br>CRYPTO_ALGOFAM_SHA3_256<br>CRYPTO_ALGOFAM_SHA3_384<br>CRYPTO_ALGOFAM_SHA3_512<br>CRYPTO_ALGOFAM_SHA3_SHAKE128<br>CRYPTO_ALGOFAM_SHA3_SHAKE256 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **CsmSignatureVerifyAlgorithmSecondaryFamilyCustom** |
|---|---|

| | | |
|---|---|---|
| **Label** | CsmSignatureVerifyAlgorithmSecondaryFamilyCustom | |
| **Description** | Name of the custom secondary algorithm family used for the crypto service. This is the name of the custom algorithm family, if CRYPTO_ALGOFAM_CUSTOM is used as CsmSignatureVerifyAlgorithmSecondaryFamily. | |
| **Multiplicity** | 0..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureVerifyCompareLength** | |
|---|---|---|
| **Label** | CsmSignatureVerifyCompareLength | |
| **Description** | Size of the input data length, for whichs signature shall be verified, in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1 | |
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureVerifyDataMaxLength** |
|---|---|
| **Label** | CsmSignatureVerifyDataMaxLength |
| **Description** | Size of the input data length, for whichs signature shall be verified, in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1 |

| | | |
|---|---|---|
| | <=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmSignatureVerifyKeyLength** | |
|---|---|---|
| **Label** | CsmSignatureVerifyKeyLength | |
| **Description** | Size of the signature verify key in bytes.<br>Range:<br>► Integer : 1 .. 4294967295 | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 1 | |
| **Range** | >=1<br><=4294967295 | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | Elektrobit | |

| **Parameter Name** | **CsmSignatureVerifyProcessing** | |
|---|---|---|
| **Label** | CsmSignatureVerifyProcessing | |
| **Description** | Determines how the interface shall be used for that primitive. Synchronous processing returns with the result while asynchronous processing returns without processing the job. The caller will be notified by the corresponding callback.<br>Range:<br>► CSM_ASYNCHRONOUS<br>► CSM_SYNCHRONOUS | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | CSM_ASYNCHRONOUS | |
| **Range** | CSM_ASYNCHRONOUS<br>CSM_SYNCHRONOUS | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

### 5.3.1.32. CsmQueues

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmQueue | 1..n | **Label:** CsmQueue<br><br>Container for configuration of a CSM queue. The container name serves as a symbolic name for the identifier of a queue configuration.<br><br>A queue has two tasks:<br><br>► queue jobs which cannot be processed since the underlying hardware is busy and<br><br>► refer to channel which shall be used |

### 5.3.1.33. CsmQueue

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmChannelRef | 1..1 |
| CsmQueueSize | 1..1 |

| **Parameter Name** | **CsmChannelRef** | |
|---|---|---|
| **Label** | CsmChannelRef | |
| **Description** | Refers to the underlying Crypto Interface channel. | |
| **Multiplicity** | 1..1 | |
| **Type** | SYMBOLIC-NAME-REFERENCE | |
| **Configuration class** | **VariantPreCompile:** | VariantPreCompile |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **CsmQueueSize** |
|---|---|
| **Label** | CsmQueueSize |
| **Description** | Size of the CsmQueue. If jobs cannot be processed by the underlying hardware since the hardware is busy, the jobs stay in the prioritized queue. If the queue is full, the next job will be rejected.<br><br>Range: |

| | |
|---|---|
| | ▶ Integer : 1 .. 4294967295 |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 1 |
| **Range** | >=1<br><=4294967295 |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | AUTOSAR_ECUC |

### 5.3.1.34. CsmEbGeneral

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CsmEbMisc | 1..1 | Configuration of miscellaneous options. |

### 5.3.1.35. CsmEbMisc

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| CsmEbAutosarApiVersion | 1..1 |
| CsmEbCorrectionCsiCsmKeyManagementCsoKeyElementGet | 1..1 |

| **Parameter Name** | **CsmEbAutosarApiVersion** |
|---|---|
| **Description** | Switches the compatibility of the Csm module API and ARXML description as specified by the configured AUTOSAR version.<br><br>▶ CSM_API_VERSION_430 = Provide and expect an API and ARXML description as specified by AUTOSAR v4.3.0. Deviations are documented in the release notes.<br>▶ CSM_API_VERSION_431 = Provide and expect an API and ARXML description as specified by AUTOSAR v4.3.1. Deviations are documented in the release notes.<br>▶ CSM_API_VERSION_EB = Provide and expect an API and ARXML description as used by EB in conjunction with CryIf modules less than version 3.0.15 and Crypto modules less than version 2.0.0. |

| | |
|---|---|
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CSM_API_VERSION_430 |
| **Range** | CSM_API_VERSION_430<br>CSM_API_VERSION_431<br>CSM_API_VERSION_EB |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **CsmEbCorrectionCsiCsmKeyManagementCsoKeyElementGet** |
|---|---|
| **Description** | Switches the implementation of the Client-Server-Operation KeyElementGet of the Client-Server-Interface CsmKeyManagement_{Config} [SWS_Csm_01905] to be compliant with the original AUTOSAR specification or to be correct respective to the specification of Csm_KeyElementGet [SWS_Csm_00959].<br>► TRUE = the correction is enabled; the AUTOSAR specification is deviated<br>► FALSE = the correction is disabled; the AUTOSAR specification is fulfilled |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |
| **Default value** | false |
| **Configuration class** | **VariantPreCompile:** VariantPreCompile |
| **Origin** | Elektrobit Automotive GmbH |

### 5.3.1.36. PublishedInformation

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| PbcfgMSupport | 1..1 |

| **Parameter Name** | **PbcfgMSupport** |
|---|---|
| **Label** | PbcfgM support |
| **Description** | Specifies whether or not the Csm can use the PbcfgM module for post-build support. |
| **Multiplicity** | 1..1 |

| | |
|---|---|
| **Type** | BOOLEAN |
| **Default value** | false |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

## 5.3.2. Application programming interface (API)

### 5.3.2.1. Type definitions

#### 5.3.2.1.1. Crypto_AlgorithmFamilyType

| | |
|---|---|
| **Purpose** | Enumeration of the algorithm family. |
| **Type** | `uint8` |

#### 5.3.2.1.2. Crypto_AlgorithmInfoType

| | | |
|---|---|---|
| **Purpose** | Structure which determines the exact algorithm. Note, not every algorithm needs to specify all fields. AUTOSAR shall only allow valid combinations. | |
| **Type** | struct | |
| **Members** | `Crypto_AlgorithmFamilyType family` | |
| | `Crypto_AlgorithmFamilyType secondaryFamily` | |
| | `uint32 keyLength` | |
| | `Crypto_AlgorithmModeType mode` | |

#### 5.3.2.1.3. Crypto_AlgorithmModeType

| | |
|---|---|
| **Purpose** | Enumeration of the algorithm mode. |

| | |
|---|---|
| **Type** | uint8 |

#### 5.3.2.1.4. Crypto_JobInfoType

| | | |
|---|---|---|
| **Purpose** | Structure which contains job information (job ID and job priority). | |
| **Type** | struct | |
| **Members** | const uint32 jobId | |
| | const uint32 jobPriority | |

#### 5.3.2.1.5. Crypto_JobPrimitiveInfoType

| | | |
|---|---|---|
| **Purpose** | Structure which contains further information, which depends on the job and the crypto primitive. | |
| **Type** | struct | |
| **Members** | const uint32 callbackId | |
| | const Crypto_PrimitiveInfoType * primitiveInfo | |
| | const uint32 secureCounterId | |
| | const uint32 cryIfKeyId | |
| | const Crypto_ProcessingType processingType | |
| | const boolean callbackUpdateNotification | |

#### 5.3.2.1.6. Crypto_JobPrimitiveInputOutputType

| | | |
|---|---|---|
| **Purpose** | Structure which contains input and output information depending on the job and the crypto primitive. | |
| **Type** | struct | |
| **Members** | const uint8 * inputPtr | |
| | uint32 inputLength | |
| | const uint8 * secondaryInputPtr | |
| | uint32 secondaryInputLength | |

| | | |
|---|---|---|
| | const uint8 * tertiaryInputPtr | |
| | uint32 tertiaryInputLength | |
| | uint8 * outputPtr | |
| | uint32 * outputLengthPtr | |
| | uint8 * secondaryOutputPtr | |
| | uint32 * secondaryOutputLength-Ptr | |
| | uint64 input64 | |
| | Crypto_VerifyResultType * veri-fyPtr | |
| | uint64 * output64Ptr | |
| | Crypto_OperationModeType mode | |

#### 5.3.2.1.7. Crypto_JobStateType

| | |
|---|---|
| **Purpose** | Enumeration of the current job state. |
| **Type** | uint8 |

#### 5.3.2.1.8. Crypto_JobType

| | | |
|---|---|---|
| **Purpose** | Structure which contains further information, which depends on the job and the crypto primitive. | |
| **Type** | struct | |
| **Members** | const uint32 jobId | |
| | Crypto_JobStateType state | |
| | Crypto_JobStateType jobState | |
| | Crypto_JobPrimitiveInputOutput-Type PrimitiveInputOutput | |
| | Crypto_JobPrimitiveInputOutput-Type jobPrimitiveInputOutput | |
| | const Crypto_JobPrimitiveInfo-Type * jobPrimitiveInfo | |
| | const Crypto_JobInfoType * jobInfo | |

| | | |
|---|---|---|
| | `uint32 cryptoKeyId` | |

#### 5.3.2.1.9. Crypto_OperationModeType

| | |
|---|---|
| **Purpose** | Enumeration which operation shall be performed. This enumeration is constructed from a bit mask, where the first bit indicates 'Start', the second 'Update' and the third 'Finish'. |
| **Type** | `uint8` |

#### 5.3.2.1.10. Crypto_PrimitiveInfoType

| | | |
|---|---|---|
| **Purpose** | Structure which contains basic information about the crypto primitive. | |
| **Type** | struct | |
| **Members** | `const uint32 resultLength` | |
| | `const Crypto_ServiceInfoType service` | |
| | `const Crypto_AlgorithmInfoType algorithm` | |

#### 5.3.2.1.11. Crypto_ProcessingType

| | |
|---|---|
| **Purpose** | Enumeration of the processing type. |
| **Type** | `uint8` |

#### 5.3.2.1.12. Crypto_ServiceInfoType

| | |
|---|---|
| **Purpose** | Enumeration of the kind of the service. |
| **Type** | `uint8` |

#### 5.3.2.1.13. Crypto_VerifyResultType

| | |
|---|---|
| **Purpose** | Enumeration of the result type of verification operations. |
| **Type** | `uint8` |

#### 5.3.2.1.14. Csm_AsymPrivateKeyArrayType

| | |
|---|---|
| **Purpose** | Maximum length in bytes of a symmetric key for all algorithms; this macro is only used by the Crypto module. |
| **Type** | `uint8[{Size}]` |
| **Description** | Maximum length in bytes of an asymmetric private key for all algorithms; this macro is only used by the Crypto module Maximum length in bytes of an asymmetric public key for all algorithms; this macro is only used by the Crypto module Array long enough to store an asymmetric private key. |

#### 5.3.2.1.15. Csm_AsymPrivateKeyType

| | | |
|---|---|---|
| **Purpose** | Structure for the private asymmetrical key. | |
| **Type** | struct | |
| **Members** | `Csm_AsymPrivateKeyArrayType data` | |
| | `uint32 length` | |

#### 5.3.2.1.16. Csm_AsymPublicKeyArrayType

| | |
|---|---|
| **Purpose** | Array long enough to store an asymmetric public key. |
| **Type** | `uint8[{Size}]` |

#### 5.3.2.1.17. Csm_AsymPublicKeyType

| | | |
|---|---|---|
| **Purpose** | Structure for the public asymmetrical key. | |
| **Type** | struct | |
| **Members** | `Csm_AsymPublicKeyArrayType data` | |
| | `uint32 length` | |

#### 5.3.2.1.18. Csm_ConfigIdType

| | |
|---|---|
| **Purpose** | Identification of a CSM service configuration via a numeric identifier, that is unique within a service. The name of a CSM service configuration, i.e. the name of the container Csm_<Service>Config, shall serve as a symbolic name for this parameter. |

| | |
|---|---|
| **Type** | `uint16` |

#### 5.3.2.1.19. Csm_ResultType

| | |
|---|---|
| **Purpose** | Csm module specific return values for use in Std_ReturnType that could occur on async. |
| **Type** | `Std_ReturnType` |

#### 5.3.2.1.20. Csm_SymKeyArrayType

| | |
|---|---|
| **Purpose** | Array long enough to store a symmetric key. |
| **Type** | `uint8[{Size}]` |

#### 5.3.2.1.21. Csm_SymKeyType

| | | |
|---|---|---|
| **Purpose** | Structure for the symmetrical key. | |
| **Type** | struct | |
| **Members** | `Csm_SymKeyArrayType data` | |
| | `uint32 length` | |

### 5.3.2.2. Macro constants

#### 5.3.2.2.1. CRYPTO_AEADDECRYPT

| | |
|---|---|
| **Purpose** | AEADDecrypt Service. |
| **Value** | 0x0006U |

#### 5.3.2.2.2. CRYPTO_AEADENCRYPT

| | |
|---|---|
| **Purpose** | AEADEncrypt Service. |
| **Value** | 0x0005U |

#### 5.3.2.2.3. CRYPTO_ALGOFAM_3DES

| | |
|---|---|
| **Purpose** | 3DES cipher. |
| **Value** | 0x0013U |

#### 5.3.2.2.4. CRYPTO_ALGOFAM_AES

| | |
|---|---|
| **Purpose** | AES cipher. |
| **Value** | 0x0014U |

#### 5.3.2.2.5. CRYPTO_ALGOFAM_BLAKE_1_256

| | |
|---|---|
| **Purpose** | BLAKE-1-256 hash. |
| **Value** | 0x000FU |

#### 5.3.2.2.6. CRYPTO_ALGOFAM_BLAKE_1_512

| | |
|---|---|
| **Purpose** | BLAKE-1-512 hash. |
| **Value** | 0x0010U |

#### 5.3.2.2.7. CRYPTO_ALGOFAM_BLAKE_2s_256

| | |
|---|---|
| **Purpose** | BLAKE-2s-256 hash. |
| **Value** | 0x0011U |

#### 5.3.2.2.8. CRYPTO_ALGOFAM_BLAKE_2s_512

| | |
|---|---|
| **Purpose** | BLAKE-2s-512 hash. |
| **Value** | 0x0012U |

#### 5.3.2.2.9. CRYPTO_ALGOFAM_BRAINPOOL

| | |
|---|---|
| **Purpose** | Brainpool elliptic curve. |

| | |
|---|---|
| **Value** | 0x0018U |

#### 5.3.2.2.10. CRYPTO_ALGOFAM_CHACHA

| | |
|---|---|
| **Purpose** | ChaCha cipher. |
| **Value** | 0x0015U |

#### 5.3.2.2.11. CRYPTO_ALGOFAM_CUSTOM

| | |
|---|---|
| **Purpose** | Custom algorithm family. |
| **Value** | 0x00FFU |

#### 5.3.2.2.12. CRYPTO_ALGOFAM_ECCNIST

| | |
|---|---|
| **Purpose** | NIST ECC elliptic curves. |
| **Value** | 0x0019U |

#### 5.3.2.2.13. CRYPTO_ALGOFAM_ECIES

| | |
|---|---|
| **Purpose** | ECIES Cipher. |
| **Value** | 0x001DU |

#### 5.3.2.2.14. CRYPTO_ALGOFAM_ED25519

| | |
|---|---|
| **Purpose** | ED22518 elliptic curve. |
| **Value** | 0x0017U |

#### 5.3.2.2.15. CRYPTO_ALGOFAM_NOT_SET

| | |
|---|---|
| **Purpose** | Algorithm family is not set. |
| **Value** | 0x0000U |

#### 5.3.2.2.16. CRYPTO_ALGOFAM_RIPEMD160

| | |
|---|---|
| **Purpose** | RIPEMD hash. |
| **Value** | 0x000EU |

#### 5.3.2.2.17. CRYPTO_ALGOFAM_RNG

| | |
|---|---|
| **Purpose** | Random Number Generator. |
| **Value** | 0x001BU |

#### 5.3.2.2.18. CRYPTO_ALGOFAM_RSA

| | |
|---|---|
| **Purpose** | RSA cipher. |
| **Value** | 0x0016U |

#### 5.3.2.2.19. CRYPTO_ALGOFAM_SECURECOUNTER

| | |
|---|---|
| **Purpose** | Secure Counter. |
| **Value** | 0x001AU |

#### 5.3.2.2.20. CRYPTO_ALGOFAM_SHA1

| | |
|---|---|
| **Purpose** | SHA1 hash. |
| **Value** | 0x0001U |

#### 5.3.2.2.21. CRYPTO_ALGOFAM_SHA2_224

| | |
|---|---|
| **Purpose** | SHA2-224 hash. |
| **Value** | 0x0002U |

#### 5.3.2.2.22. CRYPTO_ALGOFAM_SHA2_256

| | |
|---|---|
| **Purpose** | SHA2-256 hash. |

| | |
|---|---|
| **Value** | 0x0003U |

#### 5.3.2.2.23. CRYPTO_ALGOFAM_SHA2_384

| | |
|---|---|
| **Purpose** | SHA2-384 hash. |
| **Value** | 0x0004U |

#### 5.3.2.2.24. CRYPTO_ALGOFAM_SHA2_512

| | |
|---|---|
| **Purpose** | SHA2-512 hash. |
| **Value** | 0x0005U |

#### 5.3.2.2.25. CRYPTO_ALGOFAM_SHA2_512_224

| | |
|---|---|
| **Purpose** | SHA2-512/224 hash. |
| **Value** | 0x0006U |

#### 5.3.2.2.26. CRYPTO_ALGOFAM_SHA2_512_256

| | |
|---|---|
| **Purpose** | SHA2-512/256 hash. |
| **Value** | 0x0007U |

#### 5.3.2.2.27. CRYPTO_ALGOFAM_SHA3_224

| | |
|---|---|
| **Purpose** | SHA3-224 hash. |
| **Value** | 0x0008U |

#### 5.3.2.2.28. CRYPTO_ALGOFAM_SHA3_256

| | |
|---|---|
| **Purpose** | SHA3-256 hash. |
| **Value** | 0x0009U |

#### 5.3.2.2.29. CRYPTO_ALGOFAM_SHA3_384

| | |
|---|---|
| **Purpose** | SHA3-384 hash. |
| **Value** | 0x000AU |

#### 5.3.2.2.30. CRYPTO_ALGOFAM_SHA3_512

| | |
|---|---|
| **Purpose** | SHA3-512 hash. |
| **Value** | 0x000BU |

#### 5.3.2.2.31. CRYPTO_ALGOFAM_SHAKE128

| | |
|---|---|
| **Purpose** | SHAKE128 hash. |
| **Value** | 0x000CU |

#### 5.3.2.2.32. CRYPTO_ALGOFAM_SHAKE256

| | |
|---|---|
| **Purpose** | SHAKE256 hash. |
| **Value** | 0x000DU |

#### 5.3.2.2.33. CRYPTO_ALGOFAM_SIPHASH

| | |
|---|---|
| **Purpose** | SipHash. |
| **Value** | 0x001CU |

#### 5.3.2.2.34. CRYPTO_ALGOMODE_12ROUNDS

| | |
|---|---|
| **Purpose** | 12 rounds (e.g. ChaCha12). |
| **Value** | 0x000DU |

#### 5.3.2.2.35. CRYPTO_ALGOMODE_20ROUNDS

| | |
|---|---|
| **Purpose** | 20 rounds (e.g. ChaCha20). |

| | |
|---|---|
| **Value** | 0x000EU |

#### 5.3.2.2.36. CRYPTO_ALGOMODE_8ROUNDS

| | |
|---|---|
| **Purpose** | 8 rounds (e.g. ChaCha8). |
| **Value** | 0x000CU |

#### 5.3.2.2.37. CRYPTO_ALGOMODE_CBC

| | |
|---|---|
| **Purpose** | Blockmode: Cipher Block Chaining. |
| **Value** | 0x0002U |

#### 5.3.2.2.38. CRYPTO_ALGOMODE_CFB

| | |
|---|---|
| **Purpose** | Blockmode: Cipher Feedback Mode. |
| **Value** | 0x0003U |

#### 5.3.2.2.39. CRYPTO_ALGOMODE_CMAC

| | |
|---|---|
| **Purpose** | Cipher-based MAC. |
| **Value** | 0x0010U |

#### 5.3.2.2.40. CRYPTO_ALGOMODE_CTR

| | |
|---|---|
| **Purpose** | Blockmode: Counter Modex. |
| **Value** | 0x0005U |

#### 5.3.2.2.41. CRYPTO_ALGOMODE_CTRDRBG

| | |
|---|---|
| **Purpose** | Counter-based Deterministic Random Bit Generator. |
| **Value** | 0x0012U |

#### 5.3.2.2.42. CRYPTO_ALGOMODE_CUSTOM

| | |
|---|---|
| **Purpose** | Custom algorithm mode. |
| **Value** | 0x00FFU |

#### 5.3.2.2.43. CRYPTO_ALGOMODE_ECB

| | |
|---|---|
| **Purpose** | Blockmode: Electronic Code Book. |
| **Value** | 0x0001U |

#### 5.3.2.2.44. CRYPTO_ALGOMODE_GCM

| | |
|---|---|
| **Purpose** | Blockmode: Galois/Counter Mode. |
| **Value** | 0x0006U |

#### 5.3.2.2.45. CRYPTO_ALGOMODE_GMAC

| | |
|---|---|
| **Purpose** | Galois MAC. |
| **Value** | 0x0011U |

#### 5.3.2.2.46. CRYPTO_ALGOMODE_HMAC

| | |
|---|---|
| **Purpose** | Hashed-based MAC. |
| **Value** | 0x000FU |

#### 5.3.2.2.47. CRYPTO_ALGOMODE_NOT_SET

| | |
|---|---|
| **Purpose** | Algorithm key is not set. |
| **Value** | 0x0000U |

#### 5.3.2.2.48. CRYPTO_ALGOMODE_OFB

| | |
|---|---|
| **Purpose** | Blockmode: Output Feedback Mode. |

| | |
|---|---|
| **Value** | 0x0004U |

#### 5.3.2.2.49. CRYPTO_ALGOMODE_RSAES_OAEP

| | |
|---|---|
| **Purpose** | RSA Optimal Asymmetric Encryption Padding. |
| **Value** | 0x0008U |

#### 5.3.2.2.50. CRYPTO_ALGOMODE_RSAES_PKCS1_v1_5

| | |
|---|---|
| **Purpose** | RSA encryption/decryption with PKCS#1 v1.5 padding. |
| **Value** | 0x0009U |

#### 5.3.2.2.51. CRYPTO_ALGOMODE_RSASSA_PKCS1_v1_5

| | |
|---|---|
| **Purpose** | RSA signature with PKCS#1 v1.5. |
| **Value** | 0x000BU |

#### 5.3.2.2.52. CRYPTO_ALGOMODE_RSASSA_PSS

| | |
|---|---|
| **Purpose** | RSA Probabilistic Signature Scheme. |
| **Value** | 0x000AU |

#### 5.3.2.2.53. CRYPTO_ALGOMODE_SIPHASH_2_4

| | |
|---|---|
| **Purpose** | Siphash-2-4. |
| **Value** | 0x0013U |

#### 5.3.2.2.54. CRYPTO_ALGOMODE_SIPHASH_4_8

| | |
|---|---|
| **Purpose** | Siphash-4-8. |
| **Value** | 0x0014U |

#### 5.3.2.2.55. CRYPTO_ALGOMODE_XTS

| | |
|---|---|
| **Purpose** | XOR-encryption-based tweaked-codebook mode with ciphertext stealing. |
| **Value** | 0x0007U |

#### 5.3.2.2.56. CRYPTO_DECRYPT

| | |
|---|---|
| **Purpose** | Decrypt Service. |
| **Value** | 0x0004U |

#### 5.3.2.2.57. CRYPTO_ENCRYPT

| | |
|---|---|
| **Purpose** | Encrypt Service. |
| **Value** | 0x0003U |

#### 5.3.2.2.58. CRYPTO_E_BUSY

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_BUSY'. |
| **Value** | 0x0002U |

#### 5.3.2.2.59. CRYPTO_E_COUNTER_OVERFLOW

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_COUNTER_OVERFLOW'. |
| **Value** | 0x000BU |

#### 5.3.2.2.60. CRYPTO_E_ENTROPY_EXHAUSTION

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_ENTROPY_EXHAUSTION'. |
| **Value** | 0x0004U |

#### 5.3.2.2.61. CRYPTO_E_JOB_CANCELED

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_JOB_CANCELED'. |

| | |
|---|---|
| **Value** | 0x000CU |

#### 5.3.2.2.62. CRYPTO_E_KEY_NOT_AVAILABLE

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_KEY_NOT_AVAILABLE'. |
| **Value** | 0x0008U |

#### 5.3.2.2.63. CRYPTO_E_KEY_NOT_VALID

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_KEY_NOT_VALID'. |
| **Value** | 0x0009U |

#### 5.3.2.2.64. CRYPTO_E_KEY_READ_FAIL

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_KEY_READ_FAIL'. |
| **Value** | 0x0006U |

#### 5.3.2.2.65. CRYPTO_E_KEY_SIZE_MISMATCH

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_KEY_SIZE_MISMATCH'. |
| **Value** | 0x000AU |

#### 5.3.2.2.66. CRYPTO_E_KEY_WRITE_FAIL

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_KEY_WRITE_FAIL'. |
| **Value** | 0x0007U |

#### 5.3.2.2.67. CRYPTO_E_QUEUE_FULL

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_QUEUE_FULL'. |
| **Value** | 0x0005U |

#### 5.3.2.2.68. CRYPTO_E_SMALL_BUFFER

| | |
|---|---|
| **Purpose** | Crypto stack Std_ReturnType extension 'CRYPTO_E_SMALL_BUFFER'. |
| **Value** | 0x0003U |

#### 5.3.2.2.69. CRYPTO_E_VER_NOT_OK

| | |
|---|---|
| **Purpose** | The result of the verification is 'false', i.e. the two compared elements are not identical. This return code shall be given as value '1'. |
| **Value** | 0x0001U |

#### 5.3.2.2.70. CRYPTO_E_VER_OK

| | |
|---|---|
| **Purpose** | The result of the verification is 'true', i.e. the two compared elements are identical. This return code shall be given as value '0'. |
| **Value** | 0x0000U |

#### 5.3.2.2.71. CRYPTO_HASH

| | |
|---|---|
| **Purpose** | Hash Service. |
| **Value** | 0x0000U |

#### 5.3.2.2.72. CRYPTO_JOBSTATE_ACTIVE

| | |
|---|---|
| **Purpose** | Job is in the state 'active'. There was already some input or there are intermediate results. This state is reached, when the 'update' or 'start' operation finishes. |
| **Value** | 0x0001U |

#### 5.3.2.2.73. CRYPTO_JOBSTATE_IDLE

| | |
|---|---|
| **Purpose** | Job is in the state 'idle'. This state is reached after Csm_Init() or when the 'Finish' state is finished. |
| **Value** | 0x0000U |

#### 5.3.2.2.74. CRYPTO_KE_CERTIFICATE_CURRENT_TIME

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_CURRENT_TIME'. |
| **Value** | 0x0013U |

#### 5.3.2.2.75. CRYPTO_KE_CERTIFICATE_DATA

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_DATA'. |
| **Value** | 0x0000U |

#### 5.3.2.2.76. CRYPTO_KE_CERTIFICATE_EXTENSIONS

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_EXTENSIONS'. |
| **Value** | 0x001BU |

#### 5.3.2.2.77. CRYPTO_KE_CERTIFICATE_ISSUER

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_ISSUER'. |
| **Value** | 0x0017U |

#### 5.3.2.2.78. CRYPTO_KE_CERTIFICATE_PARSING_FORMAT

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_PARSING_FORMAT'. |
| **Value** | 0x0012U |

#### 5.3.2.2.79. CRYPTO_KE_CERTIFICATE_SERIALNUMBER

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_SERIALNUMBER'. |
| **Value** | 0x0015U |

#### 5.3.2.2.80. CRYPTO_KE_CERTIFICATE_SIGNATURE

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_SIGNATURE'. |

| | |
|---|---|
| **Value** | 0x001CU |

#### 5.3.2.2.81. CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_SIGNATURE_ALGORITHM'. |
| **Value** | 0x0016U |

#### 5.3.2.2.82. CRYPTO_KE_CERTIFICATE_SUBJECT

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_SUBJECT'. |
| **Value** | 0x001AU |

#### 5.3.2.2.83. CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_SUBJECT_PUBLIC_KEY'. |
| **Value** | 0x0001U |

#### 5.3.2.2.84. CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_AFTER'. |
| **Value** | 0x0019U |

#### 5.3.2.2.85. CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_VALIDITY_NOT_BEFORE'. |
| **Value** | 0x0018U |

#### 5.3.2.2.86. CRYPTO_KE_CERTIFICATE_VERSION

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CERTIFICATE_VERSION'. |
| **Value** | 0x0014U |

#### 5.3.2.2.87. CRYPTO_KE_CIPHER_2NDKEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CIPHER_2NDKEY'. |
| **Value** | 0x0007U |

#### 5.3.2.2.88. CRYPTO_KE_CIPHER_IV

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CIPHER_IV'. |
| **Value** | 0x0005U |

#### 5.3.2.2.89. CRYPTO_KE_CIPHER_KEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CIPHER_KEY'. |
| **Value** | 0x0001U |

#### 5.3.2.2.90. CRYPTO_KE_CIPHER_PROOF

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_CIPHER_PROOF'. |
| **Value** | 0x0006U |

#### 5.3.2.2.91. CRYPTO_KE_KEYDERIVATION_ALGORITHM

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYDERIVATION_ALGORITHM'. |
| **Value** | 0x000FU |

#### 5.3.2.2.92. CRYPTO_KE_KEYDERIVATION_ITERATIONS

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYDERIVATION_ITERATIONS'. |
| **Value** | 0x000EU |

#### 5.3.2.2.93. CRYPTO_KE_KEYDERIVATION_PASSWORD

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYDERIVATION_PASSWORD'. |

| | |
|---|---|
| **Value** | 0x0001U |

#### 5.3.2.2.94. CRYPTO_KE_KEYDERIVATION_SALT

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYDERIVATION_SALT'. |
| **Value** | 0x000DU |

#### 5.3.2.2.95. CRYPTO_KE_KEYEXCHANGE_ALGORITHM

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYEXCHANGE_ALGORITHM'. |
| **Value** | 0x000CU |

#### 5.3.2.2.96. CRYPTO_KE_KEYEXCHANGE_BASE

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYEXCHANGE_BASE'. |
| **Value** | 0x0008U |

#### 5.3.2.2.97. CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYEXCHANGE_OWNPUBKEY'. |
| **Value** | 0x000AU |

#### 5.3.2.2.98. CRYPTO_KE_KEYEXCHANGE_PRIVKEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYEXCHANGE_PRIVKEY'. |
| **Value** | 0x0009U |

#### 5.3.2.2.99. CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYEXCHANGE_SHAREDVALUE' (naming as intended by AUTOSAR; adjusted typo). |
| **Value** | 0x0001U |

#### 5.3.2.2.100. CRYPTO_KE_KEYGENERATE_ALGORITHM

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYGENERATE_ALGORITHM'. |
| **Value** | 0x0011U |

#### 5.3.2.2.101. CRYPTO_KE_KEYGENERATE_KEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYGENERATE_KEY'. |
| **Value** | 0x0001U |

#### 5.3.2.2.102. CRYPTO_KE_KEYGENERATE_SEED

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_KEYGENERATE_SEED'. |
| **Value** | 0x0010U |

#### 5.3.2.2.103. CRYPTO_KE_MAC_KEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_MAC_KEY'. |
| **Value** | 0x0001U |

#### 5.3.2.2.104. CRYPTO_KE_MAC_PROOF

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_MAC_PROOF'. |
| **Value** | 0x0002U |

#### 5.3.2.2.105. CRYPTO_KE_RANDOM_ALGORITHM

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_RANDOM_ALGORITHM'. |
| **Value** | 0x0004U |

#### 5.3.2.2.106. CRYPTO_KE_RANDOM_SEED_STATE

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_RANDOM_SEED_STATE'. |

| | |
|---|---|
| **Value** | 0x0003U |

#### 5.3.2.2.107. CRYPTO_KE_SIGNATURE_KEY

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CRYPTO_KE_SIGNATURE_KEY'. |
| **Value** | 0x0001U |

#### 5.3.2.2.108. CRYPTO_MACGENERATE

| | |
|---|---|
| **Purpose** | MacGenerate Service. |
| **Value** | 0x0001U |

#### 5.3.2.2.109. CRYPTO_MACVERIFY

| | |
|---|---|
| **Purpose** | MacVerify Service. |
| **Value** | 0x0002U |

#### 5.3.2.2.110. CRYPTO_OPERATIONMODE_FINISH

| | |
|---|---|
| **Purpose** | Operation Mode is 'Finish'. The calculations shall be finalized. |
| **Value** | 0x0004U |

#### 5.3.2.2.111. CRYPTO_OPERATIONMODE_SINGLECALL

| | |
|---|---|
| **Purpose** | Operation Mode is 'Single Call'. Mixture of 'Start', 'Update' and 'Finish'. |
| **Value** | 0x0007U |

#### 5.3.2.2.112. CRYPTO_OPERATIONMODE_START

| | |
|---|---|
| **Purpose** | Operation Mode is 'Start'. The job's state shall be reset, i.e. previous input data and intermediate results shall be deleted. |

| | |
|---|---|
| **Value** | 0x0001U |

#### 5.3.2.2.113. CRYPTO_OPERATIONMODE_STREAMSTART

| | |
|---|---|
| **Purpose** | Operation Mode is 'Stream Start'. Mixture of 'Start' and 'Update'. Used for streaming. |
| **Value** | 0x0003U |

#### 5.3.2.2.114. CRYPTO_OPERATIONMODE_UPDATE

| | |
|---|---|
| **Purpose** | Operation Mode is 'Update'. Used to calculate intermediate results. |
| **Value** | 0x0002U |

#### 5.3.2.2.115. CRYPTO_PROCESSING_ASYNC

| | |
|---|---|
| **Purpose** | Asynchronous job processing. |
| **Value** | 0x0000U |

#### 5.3.2.2.116. CRYPTO_PROCESSING_SYNC

| | |
|---|---|
| **Purpose** | Synchronous job processing. |
| **Value** | 0x0001U |

#### 5.3.2.2.117. CRYPTO_RANDOMGENERATE

| | |
|---|---|
| **Purpose** | RandomGenerate Service. |
| **Value** | 0x000BU |

#### 5.3.2.2.118. CRYPTO_SECCOUNTERINCREMENT

| | |
|---|---|
| **Purpose** | SecureCounterIncrement Service. |
| **Value** | 0x0009U |

#### 5.3.2.2.119. CRYPTO_SECCOUNTERREAD

| | |
|---|---|
| **Purpose** | SecureCounterRead Service. |
| **Value** | 0x000AU |

#### 5.3.2.2.120. CRYPTO_SIGNATUREGENERATE

| | |
|---|---|
| **Purpose** | SignatureGenerate Service. |
| **Value** | 0x0007U |

#### 5.3.2.2.121. CRYPTO_SIGNATUREVERIFY

| | |
|---|---|
| **Purpose** | SignatureVerify Service. |
| **Value** | 0x0008U |

#### 5.3.2.2.122. CSM_API_ENABLED_DEVERRORDETECT

| | |
|---|---|
| **Purpose** | Development Error detect enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.123. CSM_API_ENABLED_KEYMNGMNT

| | |
|---|---|
| **Purpose** | Key management APIs enabled/disabled infos. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.124. CSM_API_ENABLED_SERVICE_AEADDECRYPT

| | |
|---|---|
| **Purpose** | Service AEADDecrypt APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.125. CSM_API_ENABLED_SERVICE_AEADENCRYPT

| | |
|---|---|
| **Purpose** | Service AEADEncrypt APIs enabled/disabled info. |

| | |
|---|---|
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.126. CSM_API_ENABLED_SERVICE_ASYNCHRONOUS

| | |
|---|---|
| **Purpose** | Asynchronous service interfaces enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.127. CSM_API_ENABLED_SERVICE_DECRYPT

| | |
|---|---|
| **Purpose** | Service Decrypt APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.128. CSM_API_ENABLED_SERVICE_ENCRYPT

| | |
|---|---|
| **Purpose** | Service Encrypt APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.129. CSM_API_ENABLED_SERVICE_GENERAL

| | |
|---|---|
| **Purpose** | General services interfaces enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.130. CSM_API_ENABLED_SERVICE_HASH

| | |
|---|---|
| **Purpose** | Service Hash APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.131. CSM_API_ENABLED_SERVICE_MACGENERATE

| | |
|---|---|
| **Purpose** | Service MacGenerate APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.132. CSM_API_ENABLED_SERVICE_MACVERIFY

| | |
|---|---|
| **Purpose** | Service MacVerify APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.133. CSM_API_ENABLED_SERVICE_RANDOMGENERATE

| | |
|---|---|
| **Purpose** | Service RandomGenerate APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.134. CSM_API_ENABLED_SERVICE_SIGNATUREGENERATE

| | |
|---|---|
| **Purpose** | Service SignatureGenerate APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.135. CSM_API_ENABLED_SERVICE_SIGNATUREVERIFY

| | |
|---|---|
| **Purpose** | Service SignatureVerify APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.136. CSM_API_ENABLED_SERVICE_SYNCHRONOUS

| | |
|---|---|
| **Purpose** | Synchronous service interfaces enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.137. CSM_API_ENABLED_USEDEPRECATED

| | |
|---|---|
| **Purpose** | Deprecated APIs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.138. CSM_API_ENABLED_VERSIONINFO

| | |
|---|---|
| **Purpose** | General APIs enabled/disabled info. |

| | |
|---|---|
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.139. CSM_E_INIT_FAILED

| | |
|---|---|
| **Purpose** | Development Error to be raised if initialization of Csm module failed. |
| **Value** | 0x00007U |

#### 5.3.2.2.140. CSM_E_PARAM_HANDLE

| | |
|---|---|
| **Purpose** | Development Error to be raised if keyId or jobId of requested service is out of range. |
| **Value** | 0x00004U |

#### 5.3.2.2.141. CSM_E_PARAM_POINTER

| | |
|---|---|
| **Purpose** | Development Error to be raised if API request called with invalid parameter (Nullpointer). |
| **Value** | 0x00001U |

#### 5.3.2.2.142. CSM_E_SERVICE_NOT_IDENTICAL

| | |
|---|---|
| **Purpose** | Development Error to be raised if service of the job referenced by jobId did not match the service designated by the API function. |
| **Value** | 0x000E1U |

#### 5.3.2.2.143. CSM_E_SERVICE_NOT_STARTED

| | |
|---|---|
| **Purpose** | Development Error to be raised if requested service is not initialized. |
| **Value** | 0x00009U |

#### 5.3.2.2.144. CSM_E_UNINIT

| | |
|---|---|
| **Purpose** | Development Error to be raised if API request called before initialization of Csm module. |

| | |
|---|---|
| **Value** | 0x00005U |

#### 5.3.2.2.145. CSM_INSTANCE_ID

| | |
|---|---|
| **Purpose** | Csm instance id. |
| **Value** | 0x00U |

#### 5.3.2.2.146. CSM_JOB_COUNT

| | |
|---|---|
| **Purpose** | Number of Csm jobs. |
| **Value** | {Value} |

#### 5.3.2.2.147. CSM_KEY_COUNT

| | |
|---|---|
| **Purpose** | Number of Csm keys. |
| **Value** | {Value} |

#### 5.3.2.2.148. CSM_KEY_EMPTY

| | |
|---|---|
| **Purpose** | The value representing an empty key in Crypto_JobPrimitiveInfoType. |
| **Value** | 0xFFFFFFFFU |

#### 5.3.2.2.149. CSM_RTE_ENABLED

| | |
|---|---|
| **Purpose** | General RTE enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.150. CSM_RTE_ENABLED_KEYMNGMNT

| | |
|---|---|
| **Purpose** | Key management RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.151. CSM_RTE_ENABLED_SERVICE_AEADDECRYPT

| | |
|---|---|
| **Purpose** | Service AEADDecrypt RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.152. CSM_RTE_ENABLED_SERVICE_AEADENCRYPT

| | |
|---|---|
| **Purpose** | Service AEADEncrypt RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.153. CSM_RTE_ENABLED_SERVICE_DECRYPT

| | |
|---|---|
| **Purpose** | Service Decrypt RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.154. CSM_RTE_ENABLED_SERVICE_ENCRYPT

| | |
|---|---|
| **Purpose** | Service Encrypt RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.155. CSM_RTE_ENABLED_SERVICE_GENERAL

| | |
|---|---|
| **Purpose** | General services RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.156. CSM_RTE_ENABLED_SERVICE_HASH

| | |
|---|---|
| **Purpose** | Service Hash RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.157. CSM_RTE_ENABLED_SERVICE_MACGENERATE

| | |
|---|---|
| **Purpose** | Service MacGenerate RTEs enabled/disabled info. |

| | |
|---|---|
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.158. CSM_RTE_ENABLED_SERVICE_MACVERIFY

| | |
|---|---|
| **Purpose** | Service MacVerify RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.159. CSM_RTE_ENABLED_SERVICE_RANDOMGENERATE

| | |
|---|---|
| **Purpose** | Service RandomGenerate RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.160. CSM_RTE_ENABLED_SERVICE_SIGNATUREGENERATE

| | |
|---|---|
| **Purpose** | Service SignatureGenerate RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.161. CSM_RTE_ENABLED_SERVICE_SIGNATUREVERIFY

| | |
|---|---|
| **Purpose** | Service SignatureVerify RTEs enabled/disabled info. |
| **Value** | STD_ON or STD_OFF |

#### 5.3.2.2.162. CSM_SID_AEADDECRYPT

| | |
|---|---|
| **Purpose** | The 'Csm_AEADDecrypt' API service identifier. |
| **Value** | 0x0063U |

#### 5.3.2.2.163. CSM_SID_AEADENCRYPT

| | |
|---|---|
| **Purpose** | The 'Csm_AEADEncrypt' API service identifier. |
| **Value** | 0x0062U |

#### 5.3.2.2.164. CSM_SID_CALLBACKNOTIFICATION

| | |
|---|---|
| **Purpose** | The 'Csm_CallbackNotification' API service identifier. |
| **Value** | 0x0070U |

#### 5.3.2.2.165. CSM_SID_CANCELJOB

| | |
|---|---|
| **Purpose** | The 'Csm_CancelJob' API service identifier. |
| **Value** | 0x006FU |

#### 5.3.2.2.166. CSM_SID_CERTIFICATEPARSE

| | |
|---|---|
| **Purpose** | The 'Csm_CertificateParse' API service identifier. |
| **Value** | 0x006EU |

#### 5.3.2.2.167. CSM_SID_CERTIFICATEVERIFY

| | |
|---|---|
| **Purpose** | The 'Csm_CertificateVerify' API service identifier. |
| **Value** | 0x0074U |

#### 5.3.2.2.168. CSM_SID_DECRYPT

| | |
|---|---|
| **Purpose** | The 'Csm_Decrypt' API service identifier. |
| **Value** | 0x005FU |

#### 5.3.2.2.169. CSM_SID_ENCRYPT

| | |
|---|---|
| **Purpose** | The 'Csm_Encrypt' API service identifier. |
| **Value** | 0x005EU |

#### 5.3.2.2.170. CSM_SID_GETVERSIONINFO

| | |
|---|---|
| **Purpose** | The 'Csm_GetVersionInfo' API service identifier. |

| | |
|---|---|
| **Value** | 0x003BU |

#### 5.3.2.2.171. CSM_SID_HASH

| | |
|---|---|
| **Purpose** | The 'Csm_Hash' API service identifier. |
| **Value** | 0x005DU |

#### 5.3.2.2.172. CSM_SID_INIT

| | |
|---|---|
| **Purpose** | The 'Csm_Init' API service identifier. |
| **Value** | 0x0000U |

#### 5.3.2.2.173. CSM_SID_KEYCOPY

| | |
|---|---|
| **Purpose** | The 'Csm_KeyCopy' API service identifier. |
| **Value** | 0x0073U |

#### 5.3.2.2.174. CSM_SID_KEYDERIVE

| | |
|---|---|
| **Purpose** | The 'Csm_KeyDerive' API service identifier. |
| **Value** | 0x006BU |

#### 5.3.2.2.175. CSM_SID_KEYELEMENTCOPY

| | |
|---|---|
| **Purpose** | The 'Csm_KeyElementCopy' API service identifier. |
| **Value** | 0x0071U |

#### 5.3.2.2.176. CSM_SID_KEYELEMENTGET

| | |
|---|---|
| **Purpose** | The 'Csm_KeyElementGet' API service identifier. |
| **Value** | 0x0068U |

#### 5.3.2.2.177. CSM_SID_KEYELEMENTSET

| | |
|---|---|
| **Purpose** | The 'Csm_KeyElementSet' API service identifier. |
| **Value** | 0x0078U |

#### 5.3.2.2.178. CSM_SID_KEYEXCHANGECALCPUBVAL

| | |
|---|---|
| **Purpose** | The 'Csm_KeyExchangeCalcPubVal' API service identifier. |
| **Value** | 0x006CU |

#### 5.3.2.2.179. CSM_SID_KEYEXCHANGECALCSECRET

| | |
|---|---|
| **Purpose** | The 'Csm_KeyExchangeCalcSecret' API service identifier. |
| **Value** | 0x006DU |

#### 5.3.2.2.180. CSM_SID_KEYGENERATE

| | |
|---|---|
| **Purpose** | The 'Csm_KeyGenerate' API service identifier. |
| **Value** | 0x006AU |

#### 5.3.2.2.181. CSM_SID_KEYSETVALID

| | |
|---|---|
| **Purpose** | The 'Csm_KeySetValid' API service identifier. |
| **Value** | 0x0067U |

#### 5.3.2.2.182. CSM_SID_MACGENERATE

| | |
|---|---|
| **Purpose** | The 'Csm_MacGenerate' API service identifier. |
| **Value** | 0x0060U |

#### 5.3.2.2.183. CSM_SID_MACVERIFY

| | |
|---|---|
| **Purpose** | The 'Csm_MacVerify' API service identifier. |

| | |
|---|---|
| **Value** | 0x0061U |

#### 5.3.2.2.184. CSM_SID_MAINFUNCTION

| | |
|---|---|
| **Purpose** | The 'Csm_MainFunction' API service identifier. |
| **Value** | 0x0001U |

#### 5.3.2.2.185. CSM_SID_RANDOMGENERATE

| | |
|---|---|
| **Purpose** | The 'Csm_RandomGenerate' API service identifier. |
| **Value** | 0x0072U |

#### 5.3.2.2.186. CSM_SID_RANDOMSEED

| | |
|---|---|
| **Purpose** | The 'Csm_RandomSeed' API service identifier. |
| **Value** | 0x0069U |

#### 5.3.2.2.187. CSM_SID_SIGNATUREGENERATE

| | |
|---|---|
| **Purpose** | The 'Csm_SignatureGenerate' API service identifier. |
| **Value** | 0x0076U |

#### 5.3.2.2.188. CSM_SID_SIGNATUREVERIFY

| | |
|---|---|
| **Purpose** | The 'Csm_SignatureVerify' API service identifier. |
| **Value** | 0x0064U |

#### 5.3.2.2.189. CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE

| | |
|---|---|
| **Purpose** | Crypto stack key element 'CYRPTO_KE_KEYEXCHANGE_SHAREDVALUE' (naming as specified by AUTOSAR; including typo). |

| | |
|---|---|
| **Value** | 0x0001U |

#### 5.3.2.2.190. CsmConf_CsmJob_

| | |
|---|---|
| **Purpose** | Csm job 'CsmConf_CsmJob_{Name}'. |
| **Value** | {Name} {Value} |

#### 5.3.2.2.191. E_ENTROPY_EXHAUSTION

| | |
|---|---|
| **Purpose** | The service request failed because the entropy of random number generator is exhausted. |
| **Value** | 0x0003 |

#### 5.3.2.2.192. E_JOB_CANCELED

| | |
|---|---|
| **Purpose** | The service request failed because the job was canceled. |
| **Value** | 0x0007 |

#### 5.3.2.2.193. E_KEY_NOT_AVAILABLE

| | |
|---|---|
| **Purpose** | The service request failed because the key is not available. |
| **Value** | 0x0005 |

#### 5.3.2.2.194. E_KEY_NOT_VALID

| | |
|---|---|
| **Purpose** | The service request failed because key was not valid. |
| **Value** | 0x0006 |

#### 5.3.2.2.195. E_KEY_READ_FAIL

| | |
|---|---|
| **Purpose** | The service request failed because read access was denied. |
| **Value** | 0x0004 |

#### 5.3.2.2.196. E_SMALL_BUFFER

| | |
|---|---|
| **Purpose** | The service request failed because the provided buffer is too small to store the result. |
| **Value** | 0x0002 |

#### 5.3.2.2.197. xxCSMKEYNAMExx

| | |
|---|---|
| **Purpose** | The Csm key {CsmKeyName}. |
| **Value** | {Value} or CSM_KEY_EMPTY |

### 5.3.2.3. Objects

#### 5.3.2.3.1. Csm_JI_xxCSMJOBNAMExx

| | |
|---|---|
| **Purpose** | Configured instances of Crypto_JobInfoType referenced in configured instances of Crypto_JobType. |
| Type | `const Crypto_JobInfoType` |

#### 5.3.2.3.2. Csm_JPI_xxCSMJOBNAMExx

| | |
|---|---|
| **Purpose** | Configured instances of Crypto_JobPrimitiveInfoType referenced in configured instances of Crypto_JobType. |
| Type | `const Crypto_JobPrimitiveInfoType` |

#### 5.3.2.3.3. Csm_JobConfigurations

| | |
|---|---|
| **Purpose** | List of configured Csm jobs. |
| Type | `Crypto_JobType` |

#### 5.3.2.3.4. Csm_PI_xxCSMJOBNAMExx_xxCSMPRIMITIVENAMExx

| | |
|---|---|
| **Purpose** | Configured instances of Crypto_PrimitiveInfoType referenced in configured instances of Crypto_JobPrimitiveInfoType. |

| | |
|---|---|
| Type | const Crypto_PrimitiveInfoType |

### 5.3.2.4. Functions

#### 5.3.2.4.1. Csm_AEADDecrypt

| | | |
|---|---|---|
| **Purpose** | Uses the given data to perform an AEAD encryption and stores the ciphertext and the MAC in the memory locations pointed by the ciphertext pointer and Tag pointer. | |
| **Synopsis** | Std_ReturnType **Csm_AEADDecrypt** ( uint32 jobId , Crypto_OperationModeType mode , const uint8 * ciphertextPtr , uint32 ciphertextLength , const uint8 * associatedDataPtr , uint32 associatedDataLength , const uint8 * tagPtr , uint32 tagLength , uint8 * plaintextPtr , uint32 * plaintextLengthPtr , Crypto_VerifyResultType * verifyPtr ); | |
| **Service ID** | CSM_SID_AEADDECRYPT | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | jobId | Holds the identifier of the job using the CSM service. |
| | mode | Indicates which operation mode(s) to perfom. |
| | ciphertextPtr | Contains the pointer to the data to be decrypted. |
| | ciphertextLength | Contains the number of bytes to decrypt. |
| | associatedDataPtr | Contains the pointer to the associated data. |
| | associatedDataLength | Contains the length in bytes of the associated data. |
| | tagPtr | Contains the pointer to the Tag to be verified. |
| | tagLength | Contains the length in bytes of the Tag to be verified. |
| **Parameters (in,out)** | plaintextLengthPtr | Holds a pointer to the memory location in which the output length in bytes of the paintext is stored. On calling this function, this parameter shall contain the size of the buffer provided by plaintextPtr. When the |

| | | |
|---|---|---|
| | | request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | `plaintextPtr` | Contains the pointer to the data where the decrypted data shall be stored. |
| | `verifyPtr` | Contains the pointer to the result of the verification. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_BUSY` | request failed, service is still busy |
| | `CRYPTO_E_QUEUE_FULL` | request failed, the queue is full |
| | `CRYPTO_E_KEY_NOT_VALID` | request failed, the key's state is 'invalid' |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.2. Csm_AEADEncrypt

| | | |
|---|---|---|
| **Purpose** | Uses the given input data to perform a AEAD encryption and stores the ciphertext and the MAC in the memory locations pointed by the ciphertext pointer and Tag pointer. | |
| **Synopsis** | `Std_ReturnType` **`Csm_AEADEncrypt`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * plaintextPtr , uint32 plaintextLength , const uint8 * associatedDataPtr , uint32 associatedDataLength , uint8 * ciphertextPtr , uint32 * ciphertextLengthPtr , uint8 * tagPtr , uint32 * tagLengthPtr );` | |
| **Service ID** | CSM_SID_AEADENCRYPT | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| | `mode` | Indicates which operation mode(s) to perfom. |
| | `plaintextPtr` | Contains the pointer to the data to be encrypted. |
| | `plaintextLength` | Contains the number of bytes to encrypt. |
| | `associatedDataPtr` | Contains the pointer to the associated data. |
| | `associatedDataLength` | Contains the number of bytes of the associated data. |

| | | |
|---|---|---|
| **Parameters (in,out)** | `ciphertextLengthPtr` | Holds a pointer to the memory location in which the output length in bytes of the ciphertext is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| | `tagLengthPtr` | Holds a pointer to the memory location in which the output length in bytes of the Tag is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | `ciphertextPtr` | Contains the pointer to the data where the encrypted data shall be stored. |
| | `tagPtr` | Contains the pointer to the data where the Tag shall be stored. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_BUSY` | request failed, service is still busy |
| | `CRYPTO_E_QUEUE_FULL` | request failed, the queue is full |
| | `CRYPTO_E_KEY_NOT_VALID` | request failed, the key's state is 'invalid' |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.3. Csm_CallbackNotification

| | |
|---|---|
| **Purpose** | Notifies the CSM that a job has finished. This function is used by the underlying layer (CRYIF). |
| **Synopsis** | `void` **`Csm_CallbackNotification`** `( const Crypto_JobType * job , Std_ReturnType result );` |
| **Service ID** | CSM_SID_CALLBACKNOTIFICATION |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Reentrant |
| **Parameters (in)** | `job` — Holds a pointer to the job, which has finished. |

| | | |
|---|---|---|
| | result | Contains the result of the cryptographic operation. |

#### 5.3.2.4.4. Csm_CancelJob

| | | |
|---|---|---|
| **Purpose** | Removes the job in the Csm Queue and calls the job's callback with the result CRYPTO_E_JOB_CANCELED. It also passes the cancellation command to the CryIf to try to cancel the job in the Crypto Driver. | |
| **Synopsis** | `Std_ReturnType` **`Csm_CancelJob`** `( uint32 job , Crypto_OperationModeType mode );` | |
| **Service ID** | CSM_SID_CANCELJOB | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (in)** | job | Holds the identifier of the job to be canceled. |
| | mode | Not used, just for interface compatibility provided. |
| **Return Value** | Error value. | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |

#### 5.3.2.4.5. Csm_CertificateParse

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the certificate parse function to the CRYIF. | |
| **Synopsis** | `Std_ReturnType` **`Csm_CertificateParse`** `( uint32 keyId );` | |
| **Service ID** | CSM_SID_CERTIFICATEPARSE | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | keyId | Holds the identifier of the key to be used for the certificate parsing. |
| **Return Value** | Error value. | |
| | E_OK | Request successful |
| | E_NOT_OK | Request Failed |
| **Description** | {Reentrant, but not for same keyId} | |

#### 5.3.2.4.6. Csm_CertificateVerify

| | | |
|---|---|---|
| **Purpose** | Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId. Note: Only certificates stored in the same Crypto Driver can be verified against each other. If the key element CRYPTO_KE_CERTIFICATE_CURRENT_TIME is used for the verification of the validity period of the certificate indentified by verifyKeyId, it shall have the same format as the timestamp in the certificate. | |
| **Synopsis** | `Std_ReturnType` **`Csm_CertificateVerify`** `( uint32 keyId , uint32 verifyCryIfKeyId , Crypto_VerifyResultType * verifyPtr );` | |
| **Service ID** | CSM_SID_CERTIFICATEVERIFY | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant but not for the same cryptoKeyId | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key which shall be used to validate the certificate. |
| | `verifyCryIfKeyId` | Holds the identifier of the key containing the certificate to be verified. |
| **Parameters (out)** | `verifyPtr` | Holds a pointer to the memory location which will contain the result of the certificate verification. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |

#### 5.3.2.4.7. Csm_Decrypt

| | | |
|---|---|---|
| **Purpose** | Decrypts the given encrypted data and store the decrypted plaintext in the memory location pointed by the result pointer. | |
| **Synopsis** | `Std_ReturnType` **`Csm_Decrypt`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , uint8 * resultPtr , uint32 * resultLengthPtr );` | |
| **Service ID** | CSM_SID_DECRYPT | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| | `mode` | Indicates which operation mode(s) to perfom. |

| | | |
|---|---|---|
| | dataPtr | Contains the pointer to the data to be decrypted. |
| | dataLength | Contains the number of bytes to decrypt. |
| **Parameters (in,out)** | resultLengthPtr | Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | resultPtr | Contains the pointer to the memory location where the decrypted data shall be stored. |
| **Return Value** | Error value. | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |
| | CRYPTO_E_BUSY | request failed, service is still busy |
| | CRYPTO_E_QUEUE_FULL | request failed, the queue is full |
| | CRYPTO_E_KEY_NOT_VALID | request failed, the key's state is 'invalid' |
| | CRYPTO_E_SMALL_BUFFER | the provided buffer is too small to store the result |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.8. Csm_Encrypt

| | | |
|---|---|---|
| **Purpose** | Encrypts the given data and store the ciphertext in the memory location pointed by the result pointer. | |
| **Synopsis** | `Std_ReturnType Csm_Encrypt ( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , uint8 * resultPtr , uint32 * resultLengthPtr );` | |
| **Service ID** | CSM_SID_ENCRYPT | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | jobId | Holds the identifier of the job using the CSM service. |
| | mode | Indicates which operation mode(s) to perfom. |

| | | |
|---|---|---|
| | dataPtr | Contains the pointer to the data to be encrypted. |
| | dataLength | Contains the number of bytes to encrypt. |
| **Parameters (in,out)** | resultLengthPtr | Holds a pointer to the memory location in which the output length information is stored in bytes. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | resultPtr | Contains the pointer to the data where the encrypted data shall be stored. |
| **Return Value** | Error value. | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |
| | CRYPTO_E_BUSY | request failed, service is still busy |
| | CRYPTO_E_QUEUE_FULL | request failed, the queue is full |
| | CRYPTO_E_KEY_NOT_VALID | request failed, the key's state is 'invalid' |
| | CRYPTO_E_SMALL_BUFFER | the provided buffer is too small to store the result |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.9. Csm_GetVersionInfo

| | | |
|---|---|---|
| **Purpose** | Returns the version information of this module. | |
| **Synopsis** | `void` **`Csm_GetVersionInfo`** `( Std_VersionInfoType * versioninfo );` | |
| **Service ID** | CSM_SID_GETVERSIONINFO | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (out)** | versioninfo | Pointer to where to store the version information of this module. |

#### 5.3.2.4.10. Csm_Hash

| | |
|---|---|
| **Purpose** | Uses the given data to perform the hash calculation and stores the hash. |

| | | |
|---|---|---|
| **Synopsis** | `Std_ReturnType` **`Csm_Hash`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , uint8 * resultPtr , uint32 * resultLengthPtr );` | |
| **Service ID** | CSM_SID_HASH | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| | `mode` | Indicates which operation mode(s) to perfom. |
| | `dataPtr` | Contains the pointer to the data for which the hash shall be computed. |
| | `dataLength` | Contains the number of bytes to be hashed. |
| **Parameters (in,out)** | `resultLengthPtr` | Holds a pointer to the memory location in which the output length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | `resultPtr` | Contains the pointer to the data where the hash value shall be stored. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_BUSY` | request failed, service is still busy |
| | `CRYPTO_E_QUEUE_FULL` | request failed, the queue is full |
| | `CRYPTO_E_SMALL_BUFFER` | the provided buffer is too small to store the result |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.11. Csm_Init

| | |
|---|---|
| **Purpose** | Initializes the CSM module. |
| **Synopsis** | `void` **`Csm_Init`** `( void );` |
| **Service ID** | CSM_SID_INIT |

| | |
|---|---|
| **Sync/Async** | Synchronous |
| **Reentrancy** | Reentrant |

#### 5.3.2.4.12. Csm_KeyCopy

| | | |
|---|---|---|
| **Purpose** | This function shall copy all key elements from the source key to a target key. | |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyCopy`** `( uint32 keyId , uint32 targetKeyId );` | |
| **Service ID** | CSM_SID_KEYCOPY | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key whose key element shall be the source element. |
| | `targetKeyId` | Holds the identifier of the key whose key element shall be the destination element. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |
| | `CRYPTO_E_BUSY` | Request Failed, Crypto Driver Object is Busy |
| | `CRYPTO_E_KEY_NOT_AVAILABLE` | Request failed, the requested key element is not available |
| | `CRYPTO_E_KEY_READ_FAIL` | Request failed, not allowed to extract key element |
| | `CRYPTO_E_KEY_WRITE_FAIL` | Request failed, not allowed to write key element |
| | `CRYPTO_E_KEY_SIZE_MISMATCH` | Request failed, key element sizes are not compatible |
| **Description** | {Reentrant, but not for same keyId} | |

#### 5.3.2.4.13. Csm_KeyDerive

| | |
|---|---|
| **Purpose** | Derives a new key by using the key elements in the given key identified by the keyId. The given key contains the key elements for the password and salt. The derived key is stored in the key element with the id 1 of the key identified by targetCryptoKeyId. |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyDerive`** `( uint32 keyId , uint32 targetKeyId );` |

| | | |
|---|---|---|
| **Service ID** | CSM_SID_KEYDERIVE | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key which is used for key derivation. |
| | `targetKeyId` | Holds the identifier of the key which is used to store the derived key. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |
| | `CRYPTO_E_BUSY` | Request Failed, Crypto Driver Object is Busy |
| **Description** | {Reentrant, but not for same keyId} | |

#### 5.3.2.4.14. Csm_KeyElementCopy

| | | |
|---|---|---|
| **Purpose** | This function shall copy a key elements from one key to a target key. | |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyElementCopy`** `( uint32 keyId , uint32 keyElementId , uint32 targetKeyId , uint32 targetKeyElementId );` | |
| **Service ID** | CSM_SID_KEYELEMENTCOPY | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key whose key element shall be the source element. |
| | `keyElementId` | Holds the identifier of the key element which shall be the source for the copy operation. |
| | `targetKeyId` | Holds the identifier of the key whose key element shall be the destination element. |
| | `targetKeyElementId` | Holds the identifier of the key element which shall be the destination for the copy operation. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |
| | `CRYPTO_E_BUSY` | Request Failed, Crypto Driver Object is Busy |

| | | |
|---|---|---|
| | CRYPTO_E_KEY_NOT_AVAILABLE | Request failed, the requested key element is not available |
| | CRYPTO_E_KEY_READ_FAIL | Request failed, not allowed to extract key element |
| | CRYPTO_E_KEY_WRITE_FAIL | Request failed, not allowed to write key element |
| | CRYPTO_E_KEY_SIZE_MISMATCH | Request failed, key element sizes are not compatible |
| **Description** | {Reentrant, but not for the same keyId} | |

#### 5.3.2.4.15. Csm_KeyElementGet

| | | |
|---|---|---|
| **Purpose** | Retrieves the key element bytes from a specific key element of the key identified by the keyId and stores the key element in the memory location pointed by the key pointer. | |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyElementGet`** `( uint32 keyId , uint32 keyElementId , uint8 * keyPtr , uint32 * keyLengthPtr );` | |
| **Service ID** | CSM_SID_KEYELEMENTGET | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | keyId | Holds the identifier of the key from which a key element shall be extracted. |
| | keyElementId | Holds the identifier of the key element to be extracted. |
| **Parameters (in,out)** | keyLengthPtr | Holds a pointer to the memory location in which the output buffer length in bytes is stored. On calling this function, this parameter shall contain the buffer length in bytes of the keyPtr. When the request has finished, the actual size of the written input bytes shall be stored. |
| **Parameters (out)** | keyPtr | Holds the pointer to the memory location where the key shall be copied to. |
| **Return Value** | Error value. | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |

| | | |
|---|---|---|
| | CRYPTO_E_BUSY | Request Failed, Crypto Driver Object is Busy |
| | CRYPTO_E_KEY_NOT_AVAILABLE | request failed, the requested key element is not available |
| | CRYPTO_E_KEY_READ_FAIL | Request failed because read access was denied |
| | CRYPTO_E_SMALL_BUFFER | the provided buffer is too small to store the result |

#### 5.3.2.4.16. Csm_KeyElementSet

| | | |
|---|---|---|
| **Purpose** | Sets the given key element bytes to the key identified by keyId. | |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyElementSet`** `( uint32 keyId , uint32 keyElementId , const uint8 * keyPtr , uint32 keyLength );` | |
| **Service ID** | CSM_SID_KEYELEMENTSET | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Non Reentrant | |
| **Parameters (in)** | keyId | Holds the identifier of the key for which a new material shall be set. |
| | keyElementId | Holds the identifier of the key element to be written. |
| | keyPtr | Holds the pointer to the key element bytes to be processed. |
| | keyLength | Contains the number of key element bytes. |
| **Return Value** | Error value. | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |
| | CRYPTO_E_BUSY | Request Failed, Crypto Driver Object is Busy |
| | CRYPTO_E_KEY_WRITE_FAIL | Request failed because write access was denied |
| | CRYPTO_E_KEY_NOT_AVAILABLE | Request failed because the key is not available |
| | CRYPTO_E_KEY_SIZE_MISMATCH | Request failed, key element size does not match size of provided data |

#### 5.3.2.4.17. Csm_KeyExchangeCalcPubVal

| | | |
|---|---|---|
| **Purpose** | Calculates the public value of the current user for the key exchange and stores the public key in the memory location pointed by the public value pointer. | |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyExchangeCalcPubVal`** `( uint32 keyId , uint8 * publicValuePtr , uint32 * publicValueLengthPtr );` | |
| **Service ID** | CSM_SID_KEYEXCHANGECALCPUBVAL | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key which shall be used for the key exchange protocol. |
| **Parameters (in,out)** | `publicValueLengthPtr` | Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | `publicValuePtr` | Contains the pointer to the data where the public value shall be stored. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_KEY_NOT_VALID` | request failed, the key's state is 'invalid' |
| | `CRYPTO_E_SMALL_BUFFER` | the provided buffer is too small to store the result |
| **Description** | {Reentrant, but not for same keyId} | |

#### 5.3.2.4.18. Csm_KeyExchangeCalcSecret

| | |
|---|---|
| **Purpose** | Calculates the shared secret key for the key exchange with the key material of the key identified by the keyId and the partner public key. The shared secret key is stored as a key element in the same key. |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyExchangeCalcSecret`** `( uint32 keyId , const uint8 * partnerPublicValuePtr , uint32 partnerPublicValueLength );` |
| **Service ID** | CSM_SID_KEYEXCHANGECALCSECRET |
| **Sync/Async** | Synchronous |

| | | |
|---|---|---|
| **Reentrancy** | Reentrant but not for same keyId | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key which shall be used for the key exchange protocol. |
| | `partnerPublicValuePtr` | Holds the pointer to the memory location which contains the partner's public value. |
| | `partnerPublicValueLength` | Contains the length of the partner's public value in bytes. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |
| | `CRYPTO_E_BUSY` | Request Failed, Crypto Driver Object is Busy |
| | `CRYPTO_E_SMALL_BUFFER` | The provided buffer is too small to store the result |

#### 5.3.2.4.19. Csm_KeyGenerate

| | | |
|---|---|---|
| **Purpose** | Generates new key material and store it in the key identified by keyId. | |
| **Synopsis** | `Std_ReturnType` **`Csm_KeyGenerate`** `( uint32 keyId );` | |
| **Service ID** | CSM_SID_KEYGENERATE | |
| **Sync/Async** | Synchronous | |
| **Reentrancy** | Reentrant but not for same keyId | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key for which a new material shall be generated. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |

#### 5.3.2.4.20. Csm_KeySetValid

| | |
|---|---|
| **Purpose** | Sets the key state of the key identified by keyId to valid. |
| **Synopsis** | `Std_ReturnType` **`Csm_KeySetValid`** `( uint32 keyId );` |
| **Service ID** | CSM_SID_KEYSETVALID |
| **Sync/Async** | Synchronous |

| | | |
|---|---|---|
| **Reentrancy** | Non Reentrant | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key for which a new material shall be validated. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_BUSY` | Request Failed, Crypro Driver Object is Busy |

#### 5.3.2.4.21. Csm_MacGenerate

| | | |
|---|---|---|
| **Purpose** | Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer. | |
| **Synopsis** | `Std_ReturnType` **`Csm_MacGenerate`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , uint8 * macPtr , uint32 * macLengthPtr );` | |
| **Service ID** | CSM_SID_MACGENERATE | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| | `mode` | Indicates which operation mode(s) to perfom. |
| | `dataPtr` | Contains the pointer to the data for which the MAC shall be computed. |
| | `dataLength` | Contains the number of bytes to be hashed. |
| **Parameters (in,out)** | `macLengthPtr` | Holds a pointer to the memory location in which the output length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer provided by macPtr. When the request has finished, the actual length of the returned MAC shall be stored. |
| **Parameters (out)** | `macPtr` | Contains the pointer to the data where the MAC shall be stored. |
| **Return Value** | Error value. | |

| | | |
|---|---|---|
| | E_OK | request successful |
| | E_NOT_OK | request failed |
| | CRYPTO_E_BUSY | request failed, service is still busy |
| | CRYPTO_E_QUEUE_FULL | request failed, the queue is full |
| | CRYPTO_E_KEY_NOT_VALID | request failed, the key's state is 'invalid' |
| | CRYPTO_E_SMALL_BUFFER | the provided buffer is too small to store the result |
| **Description** | {Asynchronous or Async, dependend on the job configuration} | |

#### 5.3.2.4.22. Csm_MacVerify

| | | |
|---|---|---|
| **Purpose** | Verifies the given MAC by comparing if the MAC is generated with the given data. | |
| **Synopsis** | `Std_ReturnType` **`Csm_MacVerify`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , const uint8 * macPtr , uint32 macLength , Crypto_VerifyResultType * verifyPtr );` | |
| **Service ID** | CSM_SID_MACVERIFY | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | jobId | Indicates which operation mode(s) to perfom. |
| | mode | Indicates which operation mode(s) to perfom. |
| | dataPtr | Holds a pointer to the data for which the MAC shall be verified. |
| | dataLength | Contains the number of data bytes for which the MAC shall be verified. |
| | macPtr | Holds a pointer to the MAC to be verified. |
| | macLength | Contains the MAC length in BITS to be verified. |
| **Parameters (out)** | verifyPtr | Holds a pointer to the memory location, which will hold the result of the MAC verification. |
| **Return Value** | Error value. | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |

| | | |
|---|---|---|
| | CRYPTO_E_BUSY | request failed, service is still busy |
| | CRYPTO_E_QUEUE_FULL | request failed, the queue is full |
| | CRYPTO_E_KEY_NOT_VALID | request failed, the key's state is 'invalid' |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.23. Csm_MainFunction

| | |
|---|---|
| **Purpose** | API to be called cyclically to process the requested jobs. The Csm_MainFunction shall check the queues for jobs to pass to the underlying CRYIF. |
| **Synopsis** | `void Csm_MainFunction ( void );` |
| **Service ID** | CSM_SID_MAINFUNCTION |
| **Sync/Async** | Synchronous |
| **Reentrancy** | Non Reentrant |

#### 5.3.2.4.24. Csm_RandomGenerate

| | | |
|---|---|---|
| **Purpose** | Generate a random number and stores it in the memory location pointed by the result pointer. | |
| **Synopsis** | `Std_ReturnType Csm_RandomGenerate ( uint32 jobId , uint8 * resultPtr , uint32 * resultLengthPtr );` | |
| **Service ID** | CSM_SID_RANDOMGENERATE | |
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| **Parameters (in,out)** | `resultLengthPtr` | Holds a pointer to the memory location in which the result length in bytes is stored. On calling this function, this parameter shall contain the number of random bytes, which shall be stored to the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | `resultPtr` | Holds a pointer to the memory location which will hold the result of the random number generation. |
| **Return Value** | Error value. | |

| | | |
|---|---|---|
| | E_OK | request successful |
| | E_NOT_OK | request failed |
| | CRYPTO_E_BUSY | request failed, service is still busy |
| | CRYPTO_E_QUEUE_FULL | request failed, the queue is full |
| | CRYPTO_E_ENTROPY_EXHAUSTION | request failed, entropy of random number generator is exhausted |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.25. Csm_RandomSeed

| | | |
|---|---|---|
| **Purpose** | This function shall dispatch the random seed function to the configured crypto driver object. | |
| **Synopsis** | `Std_ReturnType` **`Csm_RandomSeed`** `( uint32 keyId , const uint8 * seedPtr , uint32 seedLength );` | |
| **Service ID** | CSM_SID_RANDOMSEED | |
| **Sync/Async** | Synchronous | |
| **Parameters (in)** | `keyId` | Holds the identifier of the key for which a new seed shall be generated. |
| | `seedPtr` | Holds a pointer to the memory location which contains the data to feed the seed. |
| | `seedLength` | Contains the length of the seed in bytes. |
| **Return Value** | Error value. | |
| | `E_OK` | Request successful |
| | `E_NOT_OK` | Request Failed |
| **Description** | {Reentrant, but not for same keyId} | |

#### 5.3.2.4.26. Csm_SignatureGenerate

| | |
|---|---|
| **Purpose** | Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer. |
| **Synopsis** | `Std_ReturnType` **`Csm_SignatureGenerate`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , uint8 * resultPtr , uint32 * resultLengthPtr );` |
| **Service ID** | CSM_SID_SIGNATUREGENERATE |

| | | |
|---|---|---|
| **Reentrancy** | Reentrant | |
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| | `mode` | Indicates which operation mode(s) to perform. |
| | `dataPtr` | Contains the pointer to the data to be signed. |
| | `dataLength` | Contains the number of bytes to sign. |
| **Parameters (in,out)** | `resultLengthPtr` | Holds a pointer to the memory location in which the output length in bytes of the signature is stored. On calling this function, this parameter shall contain the size of the buffer provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored. |
| **Parameters (out)** | `resultPtr` | Contains the pointer to the data where the signature shall be stored. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_BUSY` | request failed, service is still busy |
| | `CRYPTO_E_QUEUE_FULL` | request failed, the queue is full |
| | `CRYPTO_E_KEY_NOT_VALID` | request failed, the key's state is 'invalid' |
| | `CRYPTO_E_SMALL_BUFFER` | the provided buffer is too small to store the result |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.27. Csm_SignatureVerify

| | |
|---|---|
| **Purpose** | Verifies the given MAC by comparing if the signature is generated with the given data. |
| **Synopsis** | `Std_ReturnType` **`Csm_SignatureVerify`** `( uint32 jobId , Crypto_OperationModeType mode , const uint8 * dataPtr , uint32 dataLength , const uint8 * signaturePtr , uint32 signatureLength , Crypto_VerifyResultType * verifyPtr );` |
| **Service ID** | CSM_SID_SIGNATUREVERIFY |
| **Reentrancy** | Reentrant |

| | | |
|---|---|---|
| **Parameters (in)** | `jobId` | Holds the identifier of the job using the CSM service. |
| | `mode` | The Crypto_JobInfoType job with the corresponding jobId shall be modified in the following way:. |
| | `dataPtr` | Contains the pointer to the data to be verified. |
| | `dataLength` | Contains the number of data bytes. |
| | `signaturePtr` | Holds a pointer to the signature to be verified. |
| | `signatureLength` | Contains the signature length in bytes. |
| **Parameters (out)** | `verifyPtr` | Holds a pointer to the memory location, which will hold the result of the signature verification. |
| **Return Value** | Error value. | |
| | `E_OK` | request successful |
| | `E_NOT_OK` | request failed |
| | `CRYPTO_E_BUSY` | request failed, service is still busy |
| | `CRYPTO_E_QUEUE_FULL` | request failed, the queue is full |
| | `CRYPTO_E_KEY_NOT_VALID` | request failed, the key's state is 'invalid' |
| | `CRYPTO_E_SMALL_BUFFER` | the provided buffer is too small to store the result |
| **Description** | {Sync or Async, dependend on the job configuration} | |

#### 5.3.2.4.28. xxCSMCALLBACKNAMExx

| | |
|---|---|
| **Purpose** | Declarations of configured Csm callbacks. |
| **Synopsis** | `void` **`xxCSMCALLBACKNAMExx`** `( const Crypto_JobType * job , Std_ReturnType result );` |

## 5.3.3. Integration notes

### 5.3.3.1. Exclusive areas

This section describes the exclusive areas used by the `Csm` module.

#### 5.3.3.1.1. SCHM_CSM_EXCLUSIVE_AREA_0

| | |
|---|---|
| **Protected data structures** | All shared data that shall be protected from mutual access. |
| **Recommended locking mechanism** | This exclusive area must always be protected by a locking mechanism. The options for locking are described in the `EB tresos AutoCore Generic documentation`. Refer to the section `Mapping exclusive areas in the basic software modules` in the `Integration notes` section for details. |

### 5.3.3.2. Production errors

Production errors are not reported by the `Csm` module.

### 5.3.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section `Memory mapping and compiler abstraction` in the `Integration notes` section for details.

The following table provides the list of sections that may be mapped for this module:

| Memory section |
|---|
| CODE |
| VAR_INIT_BOOLEAN |
| VAR_INIT_UNSPECIFIED |
| CONST_UNSPECIFIED |

### 5.3.3.4. Integration requirements

**WARNING** **Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

#### 5.3.3.4.1. Csm.Req.Integration_CsmInit

| | |
|---|---|
| **Description** | Csm_Init() shall be called during the start-up procedure of the ECU before any other API of the module is called. |

#### 5.3.3.4.2. Csm.Req.Integration_UInt64_EB

| | |
|---|---|
| **Description** | If<br>► the Csm module is used within an EB tresos Studio configuration project AND<br>► the Base module is included in this an EB tresos Studio configuration project AND<br>► the Csm module configuration parameter Csm/CsmEbGeneral/CsmEb-Misc/CsmEbAutosarApiVersion is configured to CSM_API_VERSION_430 or CSM_API_VERSION_431,<br>then the Base module configuration parameter Base/BaseTypes/BaseTypes64bit shall be configured to TRUE to provide the AUTOSAR datatype 'uint64' via 'Std_Types.h'. |

#### 5.3.3.4.3. Csm.Req.Integration_UInt64_nonEB_or_nonBase

| | |
|---|---|
| **Description** | If<br>► the Csm module is NOT used within an EB tresos Studio configuration project AND<br>► the Csm module configuration parameter Csm/CsmEbGeneral/CsmEb-Misc/CsmEbAutosarApiVersion is configured to CSM_API_VERSION_430 or CSM_API_VERSION_431<br>OR<br>► the Csm module is used within an EB tresos Studio configuration project AND<br>► the Base module is NOT included in this an EB tresos Studio configuration project AND<br>► the Csm module configuration parameter Csm/CsmEbGeneral/CsmEb-Misc/CsmEbAutosarApiVersion is configured to CSM_API_VERSION_430 or CSM_API_VERSION_431,<br>then the AUTOSAR datatype 'uint64' has to be provided via 'Std_Types.h'. |

#### 5.3.3.4.4. Csm.Req.Integration_PrimitiveJob

| | |
|---|---|
| **Description** | For each job configured in Csm module a corresponding primitive has to be provided. |

#### 5.3.3.4.5. Csm.Req.Integration_Queue

| | |
|---|---|
| **Description** | For each job configured in Csm module a corresponding queue has to be provided. |

#### 5.3.3.4.6. Csm.Req.Integration_KeyRefJob

| | |
|---|---|
| **Description** | For any primitive, except Hash, a key shall be referenced by the corresponding job. That means that a dummy key shall be provided even if some drivers might not need a key for a primitive (apart from Hash), e.g. a true random number generator. |

#### 5.3.3.4.7. Csm.Req.Integration_KeyMgmt

| | |
|---|---|
| **Description** | Key management functions are only available if at least one key exists in the configuration. Otherwise, they are disabled via compiler switch and thus cannot be called. This applies to the following functions:<br>▶ Csm_KeyElementSet<br>▶ Csm_KeySetValid<br>▶ Csm_KeyElementGet<br>▶ Csm_KeyElementCopy<br>▶ Csm_KeyCopy<br>▶ Csm_KeyGenerate<br>▶ Csm_KeyDerive<br>▶ Csm_KeyExchangeCalcPubVal<br>▶ Csm_KeyExchangeCalcSecret<br>▶ Csm_CertificateParse<br>▶ Csm_CertificateVerify<br>▶ Csm_RandomSeed |

# 5.4. SecOC

## 5.4.1. Configuration parameters

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| CommonPublishedInformation | 1..1 | **Label:** Common Published Information<br>Common container, aggregated by all modules. It contains published information about vendor and versions. |
| PublishedInformation | 1..1 | **Label:** EB Published Information<br>Additional published parameters not covered by CommonPublishedInformation container. |
| SecOCGeneral | 1..1 | |
| SecOCSameBufferPduCollection | 0..n | The buffer configuration that may be used by a collection of Pdus.<br><br>The buffer can be used either by Rx PDUs or Tx PDUs, it cannot be used/ configured that both Rx and Tx PDUs can use it. |
| SecOCRxPduProcessing | 0..65535 | |
| SecOCTxPduProcessing | 0..65535 | |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| IMPLEMENTATION_CONFIG_VARIANT | 1..1 |

| Parameter Name | IMPLEMENTATION_CONFIG_VARIANT | |
|---|---|---|
| **Label** | Config Variant | |
| **Description** | Select the configuration variant. | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | VariantPostBuild | |
| **Range** | VariantPostBuild | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |

### 5.4.1.1. CommonPublishedInformation

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |

| Parameters included | |
|---|---|
| ArMajorVersion | 1..1 |
| ArMinorVersion | 1..1 |
| ArPatchVersion | 1..1 |
| SwMajorVersion | 1..1 |
| SwMinorVersion | 1..1 |
| SwPatchVersion | 1..1 |
| ModuleId | 1..1 |
| VendorId | 1..1 |
| Release | 1..1 |

| Parameter Name | ArMajorVersion |
|---|---|
| Label | AUTOSAR Major Version |
| Description | Major version number of AUTOSAR specification on which the appropriate implementation is based on. |
| Multiplicity | 1..1 |
| Type | INTEGER_LABEL |
| Default value | 4 |
| Configuration class | PublishedInformation: |
| Origin | Elektrobit Automotive GmbH |

| Parameter Name | ArMinorVersion |
|---|---|
| Label | AUTOSAR Minor Version |
| Description | Minor version number of AUTOSAR specification on which the appropriate implementation is based on. |
| Multiplicity | 1..1 |
| Type | INTEGER_LABEL |
| Default value | 3 |
| Configuration class | PublishedInformation: |
| Origin | Elektrobit Automotive GmbH |

| Parameter Name | ArPatchVersion |
|---|---|
| Label | AUTOSAR Patch Version |
| Description | Patch level version number of AUTOSAR specification on which the appropriate implementation is based on. |

| | |
|---|---|
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 0 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwMajorVersion** |
|---|---|
| **Label** | Software Major Version |
| **Description** | Major version number of the vendor specific implementation of the module. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 2 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwMinorVersion** |
|---|---|
| **Label** | Software Minor Version |
| **Description** | Minor version number of the vendor specific implementation of the module. The numbering is vendor specific. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 7 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SwPatchVersion** |
|---|---|
| **Label** | Software Patch Version |
| **Description** | Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 6 |
| **Configuration class** | **PublishedInformation:** |

| | |
|---|---|
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **ModuleId** |
|---|---|
| **Label** | Numeric Module ID |
| **Description** | Module ID of this module from Module List |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 607 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **VendorId** |
|---|---|
| **Label** | Vendor ID |
| **Description** | Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER_LABEL |
| **Default value** | 1 |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **Release** |
|---|---|
| **Label** | Release Information |
| **Multiplicity** | 1..1 |
| **Type** | STRING_LABEL |
| **Default value** | |
| **Configuration class** | **PublishedInformation:** |
| **Origin** | Elektrobit Automotive GmbH |

#### 5.4.1.2. PublishedInformation

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| PbcfgMSupport | 1..1 |

| Parameter Name | PbcfgMSupport |
|---|---|
| Label | PbcfgM support |
| Description | Specifies whether or not the SecOC can use the PbcfgM module for post-build support. |
| Multiplicity | 1..1 |
| Type | BOOLEAN |
| Default value | true |
| Configuration class | **PublishedInformation:** |
| Origin | Elektrobit Automotive GmbH |

### 5.4.1.3. SecOCGeneral

| Containers included | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCBypassAuthentication-Routine | 0..1 | **Label:** Bypass Authentication Routine<br><br>This feature provides the ability to bypass the authentication routine, when enabled, the secured PDU shall be send to the lower layer with a default value for the authentication information (authenticator/MAC + truncated freshness value).<br><br>Furthermore, when this feature is enabled during the runtime by calling the provided API, the FvM and Csm interface shall not be called.<br><br>Enable support for SecOC_BypassAuthRoutine() API.<br><br>This API can be used to enabled/disable the bypass mechanism during the runtime. |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCMainFunctionPeriodRx | 1..1 |
| SecOCMainFunctionPeriodTx | 1..1 |
| SecOCQueryFreshnessValue | 1..1 |
| SecOCVersionInfoApi | 1..1 |
| SecOcIgnoreVerificationResult | 1..1 |

| Parameters included | |
|---|---|
| SecOCDevErrorDetect | 1..1 |
| SecOCEnableForcedPassOverride | 1..1 |
| SecOCMaxAlignScalarType | 1..1 |
| SecOCVerificationStatusCallout | 0..65535 |
| SecOCMacGenerateStatusCallout | 0..65535 |
| SecOCASR403 | 1..1 |
| SecOCRteUsage | 1..1 |
| SecOCUseSecuredArea | 1..1 |
| SecOCCryptoBitLength | 1..1 |
| SecOCRelocatablePbcfgEnable | 1..1 |
| SecOCRxShapeFuncName | 0..1 |
| SecOCTxShapeFuncName | 0..1 |
| SecOCDefaultAuthenticatorValue | 0..1 |
| SecOCPropagateVerificationStatus | 1..1 |
| SecOCDataIdLength | 1..1 |
| SecOCMaxPduBufferSize | 1..1 |
| SecOCMaxIntBufferSize | 1..1 |
| SecOCCsmJobRefCallout | 0..1 |

| Parameter Name | SecOCMainFunctionPeriodRx | |
|---|---|---|
| Label | MainFunctionRx Period | |
| Description | MainFunctionRx period in seconds. | |
| Multiplicity | 1..1 | |
| Type | FLOAT | |
| Default value | 0.01 | |
| Range | >0<br><=4294967295 | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCMainFunctionPeriodTx |
|---|---|
| Label | MainFunctionTx Period |

| | |
|---|---|
| **Description** | MainFunctionTx period in seconds. |
| **Multiplicity** | 1..1 |
| **Type** | FLOAT |
| **Default value** | 0.01 |
| **Range** | >0<br>\<=4294967295 |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **SecOCQueryFreshnessValue** |
|---|---|
| **Label** | Query Freshness Value |
| **Description** | This parameter specifies how the current freshness value shall be determined.<br>► `RTE`: SecOC queries the freshness for every PDU to process using the Rte service port `RxFreshnessManagement_<SecOCFreshnessValueId>` or `TxFreshnessManagement_<SecOCFreshnessValueId>`<br>► `CFUNC`: SecOC queries the freshness for every PDU to process using the C function defined by the configuration parameter `SecOCFreshnessValue-FuncName`<br>► `NONE`: SecOC will not use freshness mechanism |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | CFUNC |
| **Range** | NONE<br>CFUNC<br>RTE |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **SecOCVersionInfoApi** |
|---|---|
| **Label** | Version Info API |
| **Description** | Enables Version Info API. |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |

| | | |
|---|---|---|
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOcIgnoreVerificationResult** | |
|---|---|---|
| **Label** | Ignore Verification Result | |
| **Description** | The result of the authentication process (e.g. MAC Verify) is ignored after the first try and the SecOC proceeds like the result was a success. The calculation of the authenticator is still done, only its result will be ignored.<br>► `TRUE`: enabled (verification result is ignored).<br>► `FALSE`: disabled (verification result is NOT ignored). | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCDevErrorDetect** | |
|---|---|---|
| **Label** | Enable Development Error Detection | |
| **Description** | **Currently not supported.**<br>Switches the Development Error Detection and Notification ON or OFF. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCEnableForcedPassOverride** |
|---|---|
| **Label** | Enable Forced Pass Override |
| **Description** | Changes the behaviour of the `SecOc_VerifyStatusOverride()` function to override the `VerifyStatus` to "Pass" and to skip the verification procedure. |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |

| | | |
|---|---|---|
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCMaxAlignScalarType** | |
|---|---|---|
| **Label** | Type with maximal alignment restrictions | |
| **Description** | The type with maximal alignment restrictions on the platform. | |
| **Multiplicity** | 1..1 | |
| **Type** | STRING | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCVerificationStatusCallout** | |
|---|---|---|
| **Label** | Callout function name | |
| **Description** | Name of a Callout function, which may be invoked on every authentication verification attempt. | |
| **Multiplicity** | 0..65535 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCMacGenerateStatusCallout** | |
|---|---|---|
| **Label** | Callout function name | |
| **Description** | Name of a Callout function, which may be invoked after the MAC Generate failed. | |
| **Multiplicity** | 0..65535 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCASR403** |
|---|---|
| **Label** | Autosar 4.0.3 PduR |
| **Description** | Specifies whether the Autosar 4.0.3 APIs or the Autosar 4.2.1 APIs shall be used for `PduR` interfaces, e.g. `SecOC_StartOfReception()`. |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCRteUsage** | |
|---|---|---|
| **Label** | Enable Rte Usage | |
| **Description** | Switches SecOC's Rte interface ON or OFF. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCUseSecuredArea** | |
|---|---|---|
| **Label** | Use secured area | |
| **Description** | Specifies whether the option to configure an area in the Authentic I-Pdu that will be the input to the Authenticator verification algorithm is enabled or not. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCCryptoBitLength** |
|---|---|
| **Label** | Crypto Bit Length |
| **Description** | Specifies, whether the length of the authenticator can be passed to the cryptographic routines in bits.<br>► `TRUE`: the length of the authenticator is passed to the cryptographic routines in bits<br>► `FALSE`: the length of the authenticator is passed to the cryptographic routines in bytes |
| **Multiplicity** | 1..1 |

| | | |
|---|---|---|
| **Type** | BOOLEAN | |
| **Default value** | true | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCRelocatablePbcfgEnable** | |
|---|---|---|
| **Description** | Enables/disable support for relocatable postbuild configuration.<br>► True: Postbuild configuration relocatable in memory.<br>► False: Postbuild configuration not relocatable in memory. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | true | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCRxShapeFuncName** | |
|---|---|---|
| **Label** | Shaping Rx function name | |
| **Description** | This parameter specifies the name of the C API function which shall be called by the SecOC to update a project specific layout of the secured PDU, which deviates from the AUTOSAR standard, to the layout defined by the AUTOSAR standard before the verification procedure is started. | |
| **Multiplicity** | 0..1 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCTxShapeFuncName** |
|---|---|
| **Label** | Shaping Tx function name |
| **Description** | This parameter specifies the name of the C API function which shall be called to modify the layout of the secured PDU before it is send to the lower layer. So the layout of a secured PDU on the bus can be adapted project specific deviating from the AUTOSAR standard. |
| **Multiplicity** | 0..1 |
| **Type** | FUNCTION-NAME |

| | | |
|---|---|---|
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCDefaultAuthenticatorValue** | |
|---|---|---|
| **Label** | Default Authenticator Value | |
| **Description** | This parameter defines the default value for the authenticator. The configured value will be set for every byte within the authenticator.<br><br>Parameter ENABLE: SecOC shall send secured messages with the default MAC, if the MAC could not be generated, i.e. Csm_MacGenerate returns something different than E_OK.<br><br>Parameter DISABLE: SecOC shall not send secured messages, if the MAC could not be generated. | |
| **Multiplicity** | 0..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=255 | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCPropagateVerificationStatus** |
|---|---|
| **Label** | Propagation of the verification status |
| **Description** | Specifies whether the option to propagate the verification status, through RTE services or C functions, is enabled or not.<br><br>► `NONE`: SecOC will not propagate the verification status<br>► `AUTOSAR`: SecOC will propagate the verification status via the AUTOSAR defined API(s)<br>► `EB_CUSTOM`: SecOC will propagate the verification status via the custom API(s)<br><br>The difference between `AUTOSAR` and `EB_CUSTOM` is in the type `SecOC_VerificationStatusType` is an additional member `verificationStatus` where the return value of the "mac verification" or "get freshness" operations is being stored. |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |

| | | |
|---|---|---|
| **Default value** | NONE | |
| **Range** | NONE | |
| | AUTOSAR | |
| | EB_CUSTOM | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCDataIdLength** |
|---|---|
| **Label** | Data ID Length |
| **Description** | This parameter defines the length in bits of the PDU Data ID.<br>► `UINT8`: PDU Data ID will have 8 bits<br>► `UINT16`: PDU Data ID will have 16 bits<br>► `UINT32`: PDU Data ID will have 32 bits |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | UINT16 |
| **Range** | UINT8 |
| | UINT16 |
| | UINT32 |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SecOCMaxPduBufferSize** |
|---|---|
| **Label** | Max buffer size for PDUs |
| **Description** | This parameter defines the maximum size of the buffer in bytes where the received and sent PDUs are stored. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SecOCMaxIntBufferSize** |
|---|---|
| **Label** | Max buffer size for internal usage |

| | |
|---|---|
| **Description** | This parameter defines the maximum size of the buffer in bytes that are used during the verification/authentication procedure. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SecOCCsmJobRefCallout** |
|---|---|
| **Label** | Callout function to obtain the Csm job ID |
| **Description** | This parameter defines the name of the callout function that shall be called by the SecOC for every Rx and Tx PDU configured in SecOCRxPduProcessing and SecOCTxPduProcessing to obtain the Csm job ID.<br><br>This function shall be called in the context of SecOC_Init() with Csm job ID extracted from the referenced SecOCRxAuthServiceConfigRef or SecOCTxAuthServiceConfigRef as the input parameter. |
| **Multiplicity** | 0..1 |
| **Type** | FUNCTION-NAME |
| **Configuration class** | **PreCompile:** VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH |

### 5.4.1.4. SecOCBypassAuthenticationRoutine

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCDefaultAuthenticationInfoValue | 1..1 |

| **Parameter Name** | **SecOCDefaultAuthenticationInfoValue** |
|---|---|
| **Label** | Default Authentication Info Value |
| **Description** | This parameter defines the default value for the authentication information. The configured value will be set for every byte within the authentication information. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Range** | >=0<br>\<=255 |

| | | |
|---|---|---|
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

### 5.4.1.5. SecOCSameBufferPduCollection

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCBufferLength | 1..1 |

| | | |
|---|---|---|
| **Parameter Name** | **SecOCBufferLength** | |
| **Label** | Buffer Length | |
| **Description** | This parameter defines the length in bytes of the buffer, which is used by the SecOC module. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br><=4294967295 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.6. SecOCRxPduProcessing

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCRxSecuredPduLayer | 1..1 | This container specifies the Pdu that is received by the SecOC module from the PduR.<br><br>There are two available possibilities to receive the data from the lower layer.<br>SecOCRxSecuredPdu - the whole content will be received within an I-PDU (Secured I-PDU)<br><br>SecOCRxSecuredPduCollection - the whole content will be received with two I-PDUs, the Authentic I-PDU (which con- |

| Containers included | | |
|---|---|---|
| | | tains the authentic data) and the Cryptographic I-PDU (which contains the cryptographic information like MAC etc) |
| SecOCRxPduSecuredArea | 0..1 | This container specifies an area in the Authentic I-Pdu that will be the input to the Authenticator verification algorithm. If this container does not exist in the configuration the complete Authentic I-Pdu will be the input to the Authenticator verification algorithm. |
| SecOCRxAuthenticPduLayer | 1..1 | This container specifies the Pdu that is transmitted by the SecOC module to the PduR after the Mac was verified. |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOcCsmMode | 1..1 |
| SecOCRxAuthServiceConfigRef | 1..1 |
| SecOCAuthInfoTxLength | 1..1 |
| SecOCDataId | 1..1 |
| SecOCFreshnessValueId | 1..1 |
| SecOCFreshnessValueLength | 1..1 |
| SecOCFreshnessValueTxLength | 1..1 |
| SecOCFreshnessValueFuncName | 1..1 |
| SecOCAuthenticationBuildAttempts | 1..1 |
| SecOCAuthenticationVerifyAttempts | 1..1 |
| SecOCVerificationStatusPropagationMode | 1..1 |
| SecOCSameBufferPduRef | 0..1 |
| SecOCUseAuthDataFreshness | 1..1 |
| SecOCAuthDataFreshnessLen | 1..1 |
| SecOCAuthDataFreshnessStartPosition | 1..1 |
| SecOCReceptionOverflowStrategy | 1..1 |
| SecOCReceptionQueueSize | 0..1 |
| SecOCRxUseShapeFunc | 1..1 |
| SecOCRxSyncPduProcessing | 1..1 |

| Parameter Name | SecOcCsmMode |
|---|---|
| **Label** | Csm operation mode |

| | | |
|---|---|---|
| **Description** | Specifies whether the Csm job is used in synchronous or asynchronous mode. | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | SYNCHRONOUS | |
| **Range** | ASYNCHRONOUS<br>SYNCHRONOUS | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| | | |
|---|---|---|
| **Parameter Name** | **SecOCRxAuthServiceConfigRef** | |
| **Label** | AuthAlgorithm | |
| **Description** | **Currently only MAC services are supported.**<br>This parameter defines the authentication algorithm used for authentication verification.<br>The value of this parameter must be a valid configuration of a MacVerify configuration in a Csm module.<br>To be able to set the authentication service reference, the configuration Enable verification must be enabled. | |
| **Multiplicity** | 1..1 | |
| **Type** | CHOICE-REFERENCE | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| | | |
|---|---|---|
| **Parameter Name** | **SecOCAuthInfoTxLength** | |
| **Label** | AuthInfoTxLength | |
| **Description** | This parameter defines the length in bits of the authentication code, which is included in the payload of the authenticated Pdu. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=1<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| Parameter Name | SecOCDataId | |
|---|---|---|
| Label | Data ID | |
| Description | This parameter defines a numerical identifier for the secured PDU. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCFreshnessValueId | |
|---|---|---|
| Label | Freshness Value ID | |
| Description | This parameter defines the ID of the Freshness value. The Freshness value might be a normal counter or a time value. If Freshness counters are used, the FreshnessValueId with the same value must have the same FreshnessValue-Length value. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=0<br>\<=65535 | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCFreshnessValueLength | |
|---|---|---|
| Label | Freshness Value Length | |
| Description | This parameter defines the complete length in bits of the Freshness Value.<br><br>As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the Freshness Value. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCFreshnessValueTxLength |
|---|---|
| Label | Truncated Freshness Value Length |

| | | |
|---|---|---|
| **Description** | This parameter defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU.<br><br>This length is specific to the least significant bits of the complete Freshness Value.<br><br>If the parameter is 0 no Freshness Value is included in the Secured I-PDU.<br><br>The Truncated Freshness Value Length must not be greater than the Freshness Value Length | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCFreshnessValueFuncName** | |
|---|---|---|
| **Label** | Freshness Value Function Name | |
| **Description** | This parameter specifies the name of the C API function which shall be called to query the freshness for the current PDU.<br><br>The called function will have the name as <function><SecOCFreshnessValueFuncName>()<function><br><br>To be able to configure the name of the C API function the Query Freshness Value must be set on `CFUNC`. | |
| **Multiplicity** | 1..1 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCAuthenticationBuildAttempts** |
|---|---|
| **Label** | AuthenticationBuildAttempts |
| **Description** | This parameter defines the number of authentication build attempts when a verification failed because the freshness value could not be obtained or the verification of the authenticator could not be performed. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 0 |
| **Range** | >=0 |

| | | |
|---|---|---|
| | <=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCAuthenticationVerifyAttempts** | |
|---|---|---|
| **Label** | AuthenticationVerifyAttempts | |
| **Description** | This parameter specifies the number of authentication verify attempts that are to be carried out when the verification of the authentication information failed for a given Secured I-PDU. If zero is set, then only one authentication verification attempt is done.<br><br>If the freshness value length is 0 and the MAC verification was executed, but the result was invalid MAC, no additional verification attempt will be executed. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Default value** | 0 | |
| **Range** | >=0 | |
| | <=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCVerificationStatusPropagationMode** | |
|---|---|---|
| **Label** | VerificationStatusPropagationMode | |
| **Description** | This parameter is used to describe the propagation of the status of each verification attempt from the SecOC module to the application.<br><br>To be able to use this feature SecOCPropagateVerificationStatus must be enabled and the RTE must be enabled or at least one callout function must be configured in the VerificationStatus Callout container. | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Range** | BOTH | |
| | FAILURE_ONLY | |
| | NONE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |

| | |
|---|---|
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **SecOCSameBufferPduRef** | |
|---|---|---|
| **Label** | SameBufferPduRef | |
| **Description** | This reference is used to collect Pdus that are using the same SecOC buffer.<br><br>The referenced buffer must be used only by Rx PDU(s). | |
| **Multiplicity** | 0..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCUseAuthDataFreshness** | |
|---|---|---|
| **Label** | Send authentic data to Freshness SWC | |
| **Description** | This parameter indicates if a part of the authentic data from the Secured PDU shall be passed on to the SWC that verifies and generates the Freshness. If it is set to `TRUE`, the values `SecOCAuthDataFreshnessStartPosition` and `SecOCAuthDataFreshnessLen` must be set to specify the bit start position and length within the Secured PDU. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCAuthDataFreshnessLen** | |
|---|---|---|
| **Label** | Authentic data freshness length | |
| **Description** | This parameter defines the length in bits the authentic data part from the Secured PDU that will be passed on to the Freshness SWC. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=64 | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| Parameter Name | SecOCAuthDataFreshnessStartPosition | |
|---|---|---|
| Label | Authentic data freshness start position | |
| Description | This parameter defines the start position in bits (uint16) of the authentic data part from the Secured PDU that shall be passed on to the Freshness SWC. The bit position starts counting from the MSB of the first byte of the PDU. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=0<br>---<br><=65535 | |
| Configuration class | **PreCompile:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCReceptionOverflowStrategy | |
|---|---|---|
| Label | Reception overflow strategy | |
| Description | This parameter specifies the overflow strategy for receiving PDUs.<br>► `QUEUE`: Subsequent received message will be queued. **Currently not supported.**<br>► `REJECT`: Subsequent received message will be discarded<br>► `REPLACE`: Subsequent received message will replace the currently processed message | |
| Multiplicity | 1..1 | |
| Type | ENUMERATION | |
| Default value | REJECT | |
| Range | REJECT<br>---<br>REPLACE | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCReceptionQueueSize |
|---|---|
| Label | Reception Queue Size |
| Description | **Currently not supported.**<br><br>This parameter defines the queue size in case the overflow strategy for receiving PDUs is set to QUEUE. |

| | | |
|---|---|---|
| **Multiplicity** | 0..1 | |
| **Type** | INTEGER | |
| **Default value** | 0 | |
| **Range** | >=1<br>---<br><=65535 | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCRxUseShapeFunc** | |
|---|---|---|
| **Label** | Fixed secured PDU layout | |
| **Description** | This parameter indicates, whether the layout shaping functionality its enabled or not for this PDU.<br><br>By enabling this parameter, the layout of the secured PDU can be updated by the SecOC callout function which name is configured in the SecOCRxShape-FuncName parameter. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCRxSyncPduProcessing** | |
|---|---|---|
| **Label** | Enable synchronous Pdu processing | |
| **Description** | This parameter indicates whether the Pdu is processed synchronously, i.e. the Pdu is processed directly without calling the main function.<br><br>Note that manually calling the main function has no effect since the Pdu processing is done within the PduR calls of SecOC interface (i.e. SecOC_RxIndication).<br><br>Synchronous Pdu processing cannot be combined with asynchronous Csm Mode. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |

| | |
|---|---|
| **Origin** | Elektrobit Automotive GmbH |

### 5.4.1.7. SecOCRxSecuredPduLayer

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCRxSecuredPdu | 1..1 | |
| SecOCRxSecuredPduCollection | 1..1 | This container specifies two Pdus that are received by the SecOC module from the PduR and a message linking between them.<br><br>SecOCRxAuthenticPdu contains the original Authentic I-PDU, i.e. the secured data, and the SecOCRxCryptographicPdu contains the Authenticator, i.e. the actual Authentication Information. |

### 5.4.1.8. SecOCRxSecuredPdu

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCRxSecuredLayerPduId | 1..1 |
| SecOCSecuredRxPduVerification | 1..1 |
| SecOCRxSecuredLayerPduRef | 1..1 |

| **Parameter Name** | **SecOCRxSecuredLayerPduId** | |
|---|---|---|
| **Label** | Secured RX PDU Handle ID | |
| **Description** | PDU identifier assigned by SecureOnboardCommunication module. Used by PduR for `SecOC_PduRRxIndication`.<br><br>The Handle-Id Wizard can be used to set this value automatically. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCSecuredRxPduVerification** |
|---|---|

| | |
|---|---|
| **Label** | Enable verification |
| **Description** | This parameter defines whether the MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification. |
| **Multiplicity** | 1..1 |
| **Type** | BOOLEAN |
| **Default value** | true |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **SecOCRxSecuredLayerPduRef** |
|---|---|
| **Label** | Secured RX PDU Reference |
| **Description** | Reference to the global Pdu holding a secured Pdu, which shall be verified by the SecOC module. |
| **Multiplicity** | 1..1 |
| **Type** | REFERENCE |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

### 5.4.1.9. SecOCRxSecuredPduCollection

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCRxAuthenticPdu | 1..1 | **Label:** Rx Authentic PDU<br><br>This container specifies the PDU that is received by the SecOC module from the lower layer, which contains the authentic data that will form with the corresponding Cryptographic I-PDU the Secured I-PDU. |
| SecOCRxCryptographicPdu | 1..1 | **Label:** Rx Cryptographic PDU<br><br>This container specifies the Cryptographic Pdu that is received by the SecOC module from the PduR. |
| SecOCUseMessageLink | 0..1 | SecOC links an Authentic I-PDU and Cryptographic I-PDU together by repeating a specific part (Message Linker) of the Authentic I-PDU in the Cryptographic I-PDU. |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCSecuredRxPduVerification | 1..1 |

| Parameter Name | SecOCSecuredRxPduVerification | |
|---|---|---|
| Label | Enable verification | |
| Description | This parameter defines whether the MAC verification shall be performed on this Secured I-PDU. If set to false, the SecOC module extracts the Authentic I-PDU from the Secured I-PDU without verification. | |
| Multiplicity | 1..1 | |
| Type | BOOLEAN | |
| Default value | true | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

### 5.4.1.10. SecOCRxAuthenticPdu

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCRxAuthenticPduId | 1..1 |
| SecOCRxAuthenticPduRef | 1..1 |

| Parameter Name | SecOCRxAuthenticPduId | |
|---|---|---|
| Label | Authentic PDU ID | |
| Description | This parameter defines the PDU identifier of the Authentic I-PDU assigned by SecOC module. Used by PduR for SecOC_PduRRxIndication. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=0<br>---<br><=65535 | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCRxAuthenticPduRef |
|---|---|

| | | |
|---|---|---|
| **Label** | Authentic PDU Reference | |
| **Description** | Reference to the global Pdu. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.11. SecOCRxCryptographicPdu

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCRxCryptographicPduId | 1..1 |
| SecOCRxCryptographicPduRef | 1..1 |

| **Parameter Name** | **SecOCRxCryptographicPduId** | |
|---|---|---|
| **Label** | Cryptographic PDU ID | |
| **Description** | This parameter defines the PDU identifier of the Cryptographic I-PDU assigned by SecOC module. Used by PduR for `SecOC_PduRRxIndication`. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCRxCryptographicPduRef** | |
|---|---|---|
| **Label** | Cryptographic PDU Reference | |
| **Description** | Reference to the global Pdu. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.12. SecOCUseMessageLink

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCMessageLinkLen | 1..1 |
| SecOCMessageLinkPos | 1..1 |

| **Parameter Name** | **SecOCMessageLinkLen** | |
|---|---|---|
| **Label** | Message Linker length | |
| **Description** | This parameter defines the length of the Message Linker inside the Authentic I-PDU in bits. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=1<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCMessageLinkPos** | |
|---|---|---|
| **Label** | Message Linker position | |
| **Description** | This parameter defines the position of the Message Linker inside the Authentic I-PDU in bits. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.13. SecOCRxPduSecuredArea

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCSecuredRxPduLength | 1..1 |

| Parameters included | |
|---|---|
| SecOCSecuredRxPduOffset | 1..1 |

| Parameter Name | SecOCSecuredRxPduLength | |
|---|---|---|
| Label | Rx PDU secured area length | |
| Description | This parameter defines the length (in bytes) of the area within the Pdu which shall be secured. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=0<br>---<br><=65535 | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCSecuredRxPduOffset | |
|---|---|---|
| Label | Rx PDU secured area Offset | |
| Description | This parameter defines the start position (offset in bytes) of the area within the Pdu which shall be secured. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=0<br>---<br><=65535 | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

### 5.4.1.14. SecOCRxAuthenticPduLayer

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCPduType | 1..1 |
| SecOCRxAuthenticLayerPduRef | 1..1 |

| Parameter Name | SecOCPduType |
|---|---|

| | |
|---|---|
| **Label** | PduR API Type |
| **Description** | This parameter defines API Type to use for communication with PduR.<br>► `SECOC_IFPDU`: SECOC_IFPDU Interface communication API<br>► `SECOC_TPPDU`: SECOC_TPPDU Transport Protocol communication API |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | SECOC_IFPDU |
| **Range** | SECOC_IFPDU<br>SECOC_TPPDU |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

| | |
|---|---|
| **Parameter Name** | **SecOCRxAuthenticLayerPduRef** |
| **Label** | Authentic RX PDU Reference |
| **Description** | Reference to the global Pdu holding an authenticated Pdu. |
| **Multiplicity** | 1..1 |
| **Type** | REFERENCE |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

### 5.4.1.15. SecOCTxPduProcessing

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCTxSecuredPduLayer | 1..1 | This container specifies the Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated.<br>There are two available possibilities to send the data to the lower layer.<br>SecOCTxSecuredPdu - the whole content will be send within an I-PDU (Secured I-PDU)<br>SecOCTxSecuredPduCollection - the whole content will be send with two I-PDUs, the Authentic I-PDU (which contains |

| Containers included | | |
|---|---|---|
| | | the authentic data) and the Cryptographic I-PDU (which contains the cryptographic information like MAC etc) |
| SecOCTxPduSecuredArea | 0..1 | This container specifies an area in the Authentic I-Pdu that will be the input to the Authenticator generation algorithm. If this container does not exist in the configuration the complete Authentic I-Pdu will be the input to the Authenticator generation algorithm. |
| SecOCTxAuthenticPduLayer | 1..1 | This container specifies the Authetic Pdu that is received by the SecOC module from the PduR based on this the Secured Pdu is generated. |

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOcCsmMode | 1..1 |
| SecOCTxAuthServiceConfigRef | 1..1 |
| SecOCAuthInfoTxLength | 1..1 |
| SecOCDataId | 1..1 |
| SecOCFreshnessValueId | 1..1 |
| SecOCFreshnessValueLength | 1..1 |
| SecOCFreshnessValueTxLength | 1..1 |
| SecOCFreshnessValueFuncName | 1..1 |
| SecOCSecuredPDUTransmittedFuncName | 1..1 |
| SecOCAuthenticationBuildAttempts | 1..1 |
| SecOCMacGenerateStatusPropagationMode | 1..1 |
| SecOCSameBufferPduRef | 0..1 |
| SecOCProvideTxTruncatedFreshnessValue | 1..1 |
| SecOCUseTxConfirmation | 0..1 |
| SecOCTxConfirmationTimeout | 1..1 |
| SecOCTxUseShapeFunc | 1..1 |
| SecOCTxSyncPduProcessing | 1..1 |

| Parameter Name | SecOcCsmMode |
|---|---|
| **Label** | Csm operation mode |
| **Description** | Specifies whether the Csm job is used in synchronous or asynchronous mode. |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | SYNCHRONOUS | |
| **Range** | ASYNCHRONOUS<br>SYNCHRONOUS | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCTxAuthServiceConfigRef** | |
|---|---|---|
| **Label** | Authentication Algorithm | |
| **Description** | **Currently only MAC services are supported.**<br>This parameter defines the authentication algorithm used for authentication generation.<br>The value of this parameter must be a valid configuration of a MacGenerate configuration in a Csm module. | |
| **Multiplicity** | 1..1 | |
| **Type** | CHOICE-REFERENCE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCAuthInfoTxLength** | |
|---|---|---|
| **Label** | AuthInfoTxLength | |
| **Description** | This parameter defines the length in bits of the authentication code, which is included in the payload of the authenticated Pdu. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=1<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCDataId** |
|---|---|
| **Label** | Data ID |

| | | |
|---|---|---|
| **Description** | This parameter defines a numerical identifier for the secured PDU. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCFreshnessValueId** | |
|---|---|---|
| **Label** | Freshness Value ID | |
| **Description** | This parameter defines the ID of the Freshness value. The Freshness value might be a normal counter or a time value. If Freshness counters are used, the FreshnessValueId with the same value must have the same FreshnessValue-Length. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCFreshnessValueLength** | |
|---|---|---|
| **Label** | Freshness Value Length | |
| **Description** | This parameter defines the complete length in bits of the Freshness Value.<br><br>As long as the key doesn't change the counter shall not overflow. The length of the counter shall be determined based on the expected life time of the corresponding key and frequency of usage of the Freshness Value. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCFreshnessValueTxLength** |
|---|---|
| **Label** | Truncated Freshness Value Length |
| **Description** | This parameter defines the length in bits of the Freshness Value to be included in the payload of the Secured I-PDU. |

| | | |
|---|---|---|
| | This length is specific to the least significant bits of the complete Freshness Value.<br><br>If the parameter is 0 no Freshness Value is included in the Secured I-PDU.<br><br>The Truncated Freshness Value Length must not be greater than the Freshness Value Length | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCFreshnessValueFuncName** | |
|---|---|---|
| **Label** | Freshness Value Function Name | |
| **Description** | This parameter specifies the name of the C API function which shall be called to query the freshness for the current PDU.<br><br>The called function will have the name as <function><SecOCFreshnessValueFuncName>()<function><br><br>To be able to configure the name of the C API function the Query Freshness Value must be set on `CFUNC`. | |
| **Multiplicity** | 1..1 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCSecuredPDUTransmittedFuncName** | |
|---|---|---|
| **Label** | SecuredPDUTransmitted function name | |
| **Description** | This parameter specifies the name of the C API function which shall be called after a Secured I-PDU has been started for transmission. | |
| **Multiplicity** | 1..1 | |
| **Type** | FUNCTION-NAME | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCAuthenticationBuildAttempts** |
|---|---|

| | |
|---|---|
| **Label** | AuthenticationBuildAttempts |
| **Description** | This parameter defines the number of authentication build attempts when an authentication failed because the freshness value could not be obtained or the generation of the authenticator could not be performed. |
| **Multiplicity** | 1..1 |
| **Type** | INTEGER |
| **Default value** | 0 |
| **Range** | >=0<br>\<=65535 |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | AUTOSAR_ECUC |

| **Parameter Name** | **SecOCMacGenerateStatusPropagationMode** |
|---|---|
| **Label** | MacGenerateStatusPropagationMode |
| **Description** | This parameter is used to describe the propagation of the status from the SecOC module to the application.<br>► `BOTH`: SecOC will propagate both negative and posivite status of the MAC Generate service<br>► `FAILURE_ONLY`: SecOC will propagate the status only when the MAC Generate service failed<br>► `NONE`: SecOC will not propagate the status of the MAC Generate service<br>To be able to use this feature the RTE must be enabled or at least one callout function must be configured in the MacGenerateStatus Callout container. |
| **Multiplicity** | 1..1 |
| **Type** | ENUMERATION |
| **Default value** | NONE |
| **Range** | BOTH<br>FAILURE_ONLY<br>NONE |
| **Configuration class** | **VariantPostBuild:** VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH |

| **Parameter Name** | **SecOCSameBufferPduRef** |
|---|---|
| **Label** | SameBufferPduRef |

| | | |
|---|---|---|
| **Description** | This reference is used to collect Pdus that are using the same SecOC buffer.<br><br>The referenced buffer must be used only by Tx PDU(s). | |
| **Multiplicity** | 0..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCProvideTxTruncatedFreshnessValue** | |
|---|---|---|
| **Label** | Provide Truncated Freshness Value | |
| **Description** | This parameter specifies if the Tx query freshness function provides the truncated freshness info instead of generating this by SecOC. In this case, SecOC shall add this data to the Authentic PDU instead of truncating the freshness value. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCUseTxConfirmation** | |
|---|---|---|
| **Label** | Use TxConfirmation | |
| **Description** | **Currently not supported.** The function `SecOC_SPduTxConfirmation` will be enabled by default when the freshness values functions are used (Query Freshness Value != NONE).<br><br>This parameter indicates indicates if the function `SecOC_SPduTxConfirmation` shall be called for this PDU. | |
| **Multiplicity** | 0..1 | |
| **Type** | BOOLEAN | |
| **Default value** | true | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCTxConfirmationTimeout** |
|---|---|
| **Label** | Timeout period for TxConfirmation |
| **Description** | Period in seconds for TxConfirmation timeout. |

| | | |
|---|---|---|
| | If the value is 0, the timeout feature will be disabled.<br><br>If the value is different than 0, the timeout value must be equal or greater than Tx main period. | |
| **Multiplicity** | 1..1 | |
| **Type** | FLOAT | |
| **Default value** | 0 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCTxUseShapeFunc** | |
|---|---|---|
| **Label** | Fixed secured PDU layout | |
| **Description** | This parameter indicates, whether the layout shaping functionality its enabled or not for this PDU.<br><br>By enabling this parameter, the secured PDU layout can be updated by the SecOC callout function which name is configured in the SecOCTxShapeFuncName parameter. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | Elektrobit Automotive GmbH | |

| **Parameter Name** | **SecOCTxSyncPduProcessing** | |
|---|---|---|
| **Label** | Enable synchronous Pdu processing | |
| **Description** | This parameter indicates whether the Pdu is processed synchronously, i.e. the Pdu is processed directly without calling the main function.<br><br>Note that manually calling the main function has no effect since the Pdu processing is done within the PduR calls of SecOC interface (i.e. SecOC_Transmit).<br><br>Synchronous Pdu processing cannot be combined with asynchronous Csm Mode. | |
| **Multiplicity** | 1..1 | |
| **Type** | BOOLEAN | |
| **Default value** | false | |
| **Configuration class** | **PreCompile:** | VariantPostBuild |

| | |
|---|---|
| **Origin** | Elektrobit Automotive GmbH |

### 5.4.1.16. SecOCTxSecuredPduLayer

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCTxSecuredPdu | 1..1 | |
| SecOCTxSecuredPduCollection | 1..1 | This container specifies the Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated. Two separate Pdus are transmitted to the PduR: Authentic I-PDU and Cryptographic I-PDU. |

### 5.4.1.17. SecOCTxSecuredPdu

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCTxSecuredLayerPduId | 1..1 |
| SecOCTxSecuredLayerPduRef | 1..1 |

| **Parameter Name** | **SecOCTxSecuredLayerPduId** | |
|---|---|---|
| **Label** | Secured TX PDU Handle ID | |
| **Description** | PDU identifier assigned by SecureOnboardCommunication module. Used by PduR for confirmation (SecOC_PduRTxConfirmation) and for TriggerTransmit.<br><br>The Handle-Id Wizard can be used to set this value automatically. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCTxSecuredLayerPduRef** |
|---|---|
| **Label** | Secured TX PDU Reference |
| **Description** | Reference to the global Pdu, which holds the secured Pdu. |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.18. SecOCTxSecuredPduCollection

| **Containers included** | | |
|---|---|---|
| **Container name** | **Multiplicity** | **Description** |
| SecOCTxAuthenticPdu | 1..1 | **Label:** Tx Authentic PDU<br><br>This container specifies the PDU that is send by the SecOC module to the lower layer, which contains the authentic data that is forming with the corresponding Cryptographic I-PDU the Secured I-PDU. |
| SecOCTxCryptographicPdu | 1..1 | **Label:** Tx Cryptographic PDU<br><br>This container specifies the Cryptographic Pdu that is transmitted by the SecOC module to the PduR after the Mac was generated. |
| SecOCUseMessageLink | 0..1 | SecOC links an Authentic I-PDU and Cryptographic I-PDU together by repeating a specific part (Message Linker) of the Authentic I-PDU in the Cryptographic I-PDU. |

### 5.4.1.19. SecOCTxAuthenticPdu

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCTxAuthenticPduId | 1..1 |
| SecOCTxAuthenticPduRef | 1..1 |

| **Parameter Name** | **SecOCTxAuthenticPduId** |
|---|---|
| **Label** | Authentic PDU ID |
| **Description** | This parameter defines the PDU identifier of the Authentic I-PDU assigned by SecOC module. Used by PduR for confirmation (`SecOC_PduRTxConfirmation`) and for TriggerTransmit. |

| | | |
|---|---|---|
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCTxAuthenticPduRef** | |
|---|---|---|
| **Label** | Authentic PDU Reference | |
| **Description** | Reference to the global Pdu. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.20. SecOCTxCryptographicPdu

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCTxCryptographicPduId | 1..1 |
| SecOCTxCryptographicPduRef | 1..1 |

| **Parameter Name** | **SecOCTxCryptographicPduId** | |
|---|---|---|
| **Label** | Cryptographic PDU ID | |
| **Description** | This parameter defines the PDU identifier of the Cryptographic I-PDU assigned by SecOC module. Used by PduR for confirmation (SecOC_PduRTxConfirmation) and for TriggerTransmit. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| Parameter Name | SecOCTxCryptographicPduRef | |
|---|---|---|
| Label | Cryptographic PDU Reference | |
| Description | Reference to the global Pdu. | |
| Multiplicity | 1..1 | |
| Type | REFERENCE | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

### 5.4.1.21. SecOCUseMessageLink

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCMessageLinkLen | 1..1 |
| SecOCMessageLinkPos | 1..1 |

| Parameter Name | SecOCMessageLinkLen | |
|---|---|---|
| Label | Message Linker length | |
| Description | This parameter defines the length of the Message Linker inside the Authentic I-PDU in bits. | |
| Multiplicity | 1..1 | |
| Type | INTEGER | |
| Range | >=1<br>\<=65535 | |
| Configuration class | **VariantPostBuild:** | VariantPostBuild |
| Origin | AUTOSAR_ECUC | |

| Parameter Name | SecOCMessageLinkPos |
|---|---|
| Label | Message Linker position |
| Description | This parameter defines the position of the Message Linker inside the Authentic I-PDU in bits. |
| Multiplicity | 1..1 |
| Type | INTEGER |
| Range | >=0 |

| | | |
|---|---|---|
| | <=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.22. SecOCTxPduSecuredArea

| **Parameters included** | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCSecuredTxPduLength | 1..1 |
| SecOCSecuredTxPduOffset | 1..1 |

| **Parameter Name** | **SecOCSecuredTxPduLength** | |
|---|---|---|
| **Label** | Tx PDU secured area length | |
| **Description** | This parameter defines the length (in bytes) of the area within the Pdu which shall be secured. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCSecuredTxPduOffset** | |
|---|---|---|
| **Label** | Tx PDU secured area Offset | |
| **Description** | This parameter defines the start position (offset in bytes) of the area within the Pdu which shall be secured. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Range** | >=0<br>---<br><=65535 | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

### 5.4.1.23. SecOCTxAuthenticPduLayer

| Parameters included | |
|---|---|
| **Parameter name** | **Multiplicity** |
| SecOCPduType | 1..1 |
| SecOCTxAuthenticLayerPduId | 1..1 |
| SecOCTxAuthenticLayerPduRef | 1..1 |

| **Parameter Name** | **SecOCPduType** | |
|---|---|---|
| **Label** | PduR API Type | |
| **Description** | This parameter defines API Type to use for communication with PduR.<br>► `SECOC_IFPDU`: SECOC_IFPDU Interface communication API<br>► `SECOC_TPPDU`: SECOC_TPPDU Transport Protocol communication API | |
| **Multiplicity** | 1..1 | |
| **Type** | ENUMERATION | |
| **Default value** | SECOC_IFPDU | |
| **Range** | SECOC_IFPDU<br>SECOC_TPPDU | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCTxAuthenticLayerPduId** | |
|---|---|---|
| **Label** | Authentic TX PDU Handle ID | |
| **Description** | PDU identifier assigned by SecureOnboardCommunication module. Used by PduR for SecOC_PduRTransmit.<br><br>The Handle-Id Wizard can be used to set this value automatically. | |
| **Multiplicity** | 1..1 | |
| **Type** | INTEGER | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

| **Parameter Name** | **SecOCTxAuthenticLayerPduRef** |
|---|---|
| **Label** | Authentic TX PDU Reference |

| | | |
|---|---|---|
| **Description** | Reference to the global Pdu holding the authentic Pdu, for which the SecOC module shall generate an authenticator. | |
| **Multiplicity** | 1..1 | |
| **Type** | REFERENCE | |
| **Configuration class** | **VariantPostBuild:** | VariantPostBuild |
| **Origin** | AUTOSAR_ECUC | |

## 5.4.2. Application programming interface (API)

### 5.4.2.1. Type definitions

#### 5.4.2.1.1. SecOC_MacGenerateStatusType

| | | |
|---|---|---|
| **Purpose** | Data structure to bundle the status of a MAC generate attempt for a specific Freshness Value and Data ID. | |
| **Type** | struct | |
| **Members** | `uint16 freshnessValueID` | Identifier of the Freshness Value which resulted in the Verification Result. |
| | `Std_ReturnType macGenerateStatus` | Result of the MAC Generate procedure. |
| | `SecOC_DataIdLengthType secOCDataId` | Identifier for the Secured I-PDU. |

#### 5.4.2.1.2. SecOC_StateType

| | |
|---|---|
| **Purpose** | States of the SecOC module. |
| **Type** | `uint8` |
| **Description** | Range: SECOC_UNINIT, SECOC_INIT. |

#### 5.4.2.1.3. SecOC_VerificationResultType

| | |
|---|---|
| **Purpose** | Type, to indicate verification results. |

| | |
|---|---|
| **Type** | `uint8` |
| **Description** | Range: SECOC_VERIFICATIONSUCCESS, SECOC_VERIFICATIONFAILURE, SECOC_FRESHNESSFAILURE, SECOC_AUTHENTICATIONBUILDFAILURE, SECOC_NO_VERIFICATION, SECOC_MACSERVICEFAILURE. |

#### 5.4.2.1.4. SecOC_VerificationStatusType

| | | |
|---|---|---|
| **Purpose** | Data structure to bundle the status of a verification attempt for a specific Freshness Value and Data ID. | |
| **Type** | struct | |
| **Members** | `uint16 freshnessValueID` | Identifier of the Freshness Value which resulted in the Verification Result. |
| | `SecOC_VerificationResultType verificationStatus` | Result of verification attempt. |
| | `SecOC_DataIdLengthType secOCDataId` | Identifier for the Secured I-PDU. |
| | `Std_ReturnType verificationReturn` | Return of verification attempt (available only if SecOCPropagateVerificationStatus is set to EB_CUSTOM). |

### 5.4.2.2. Macro constants

#### 5.4.2.2.1. SECOC_AR_RELEASE_MAJOR_VERSION

| | |
|---|---|
| **Purpose** | AUTOSAR release major version. |
| **Value** | 4U |

#### 5.4.2.2.2. SECOC_AR_RELEASE_MINOR_VERSION

| | |
|---|---|
| **Purpose** | AUTOSAR release minor version. |
| **Value** | 3U |

#### 5.4.2.2.3. SECOC_AR_RELEASE_REVISION_VERSION

| | |
|---|---|
| **Purpose** | AUTOSAR release revision version. |

| | |
|---|---|
| **Value** | 0U |

#### 5.4.2.2.4. SECOC_AUTHENTICATIONBUILDFAILURE

| | |
|---|---|
| **Purpose** | Authentication could not be built. Freshness attempt or authentication calculation failure. |
| **Value** | SECOC_FRESHNESSFAILURE |

#### 5.4.2.2.5. SECOC_E_BUSY

| | |
|---|---|
| **Purpose** | Return value if the "get freshness value" service is currently busy. |
| **Value** | 2U |

#### 5.4.2.2.6. SECOC_E_NOT_OK

| | |
|---|---|
| **Purpose** | Return value for an unsuccessful "get freshness value" request. |
| **Value** | 1U |

#### 5.4.2.2.7. SECOC_E_OK

| | |
|---|---|
| **Purpose** | Return value for a successful "get freshness value" request. |
| **Value** | 0U |

#### 5.4.2.2.8. SECOC_FRESHNESSFAILURE

| | |
|---|---|
| **Purpose** | Verification not successful because of wrong freshness value. |
| **Value** | 2U |

#### 5.4.2.2.9. SECOC_FRESHNESS_CFUNC

| | |
|---|---|
| **Purpose** | SecOC queries the freshness for every PDU to process using the C function defined by the configuration parameter SecOCFreshnessValueFuncName. |
| **Value** | 2U |

#### 5.4.2.2.10. SECOC_FRESHNESS_NONE

| | |
|---|---|
| **Purpose** | SecOC does not queries the freshness. |
| **Value** | 0U |

#### 5.4.2.2.11. SECOC_FRESHNESS_RTE

| | |
|---|---|
| **Purpose** | SecOC queries the freshness for every PDU to process using the Rte service port FreshnessManagement. |
| **Value** | 1U |

#### 5.4.2.2.12. SECOC_GET_RX_FRESHNESS_AUTHDATA_FUNC_TYPE

| | |
|---|---|
| **Purpose** | Macro which defines that the GetTxFreshnessTruncData function shall be used to obtain the freshness value. |
| **Value** | 1U |

#### 5.4.2.2.13. SECOC_GET_RX_FRESHNESS_FUNC_TYPE

| | |
|---|---|
| **Purpose** | Macro which defines that the GetTxFreshness function shall be used to obtain the freshness value. |
| **Value** | 0U |

#### 5.4.2.2.14. SECOC_GET_TX_FRESHNESS_FUNC_TYPE

| | |
|---|---|
| **Purpose** | Macro which defines that the GetTxFreshness function shall be used to obtain the freshness value. |
| **Value** | 0U |

#### 5.4.2.2.15. SECOC_GET_TX_FRESHNESS_TRUNCDATA_FUNC_TYPE

| | |
|---|---|
| **Purpose** | Macro which defines that the GetTxFreshnessTruncData function shall be used to obtain the freshness value. |
| **Value** | 1U |

#### 5.4.2.2.16. SECOC_INIT

| | |
|---|---|
| **Purpose** | SecOC module is initialized. |
| **Value** | 1U |

#### 5.4.2.2.17. SECOC_INSTANCE_ID

| | |
|---|---|
| **Purpose** | Id of instance of SecOC. |
| **Value** | 0U |

#### 5.4.2.2.18. SECOC_MACSERVICEFAILURE

| | |
|---|---|
| **Purpose** | Verification not successful because of wrong freshness value. |
| **Value** | 8U |

#### 5.4.2.2.19. SECOC_MODULE_ID

| | |
|---|---|
| **Purpose** | AUTOSAR module identification. |
| **Value** | 607U |

#### 5.4.2.2.20. SECOC_NO_VERIFICATION

| | |
|---|---|
| **Purpose** | Verification has been skipped. |
| **Value** | 4U |

#### 5.4.2.2.21. SECOC_STATUS_PROP_BOTH

| | |
|---|---|
| **Purpose** | Defines, that Both 'True' and 'False' AuthenticationStatus is propagated. |
| **Value** | 2U |

#### 5.4.2.2.22. SECOC_STATUS_PROP_FAILURE_ONLY

| | |
|---|---|
| **Purpose** | Defines, that Only 'False' AuthenticationStatus is propagated. |

| | |
|---|---|
| **Value** | 1U |

#### 5.4.2.2.23. SECOC_STATUS_PROP_NONE

| | |
|---|---|
| **Purpose** | Defines, that No AuthenticationStatus is propagated. |
| **Value** | 0U |

#### 5.4.2.2.24. SECOC_SW_MAJOR_VERSION

| | |
|---|---|
| **Purpose** | AUTOSAR module major version. |
| **Value** | 2U |

#### 5.4.2.2.25. SECOC_SW_MINOR_VERSION

| | |
|---|---|
| **Purpose** | AUTOSAR module minor version. |
| **Value** | 7U |

#### 5.4.2.2.26. SECOC_SW_PATCH_VERSION

| | |
|---|---|
| **Purpose** | AUTOSAR module patch version. |
| **Value** | 6U |

#### 5.4.2.2.27. SECOC_UNINIT

| | |
|---|---|
| **Purpose** | SecOC module is not initialized. |
| **Value** | 0U |

#### 5.4.2.2.28. SECOC_VENDOR_ID

| | |
|---|---|
| **Purpose** | AUTOSAR vendor identification: Elektrobit Automotive GmbH. |
| **Value** | 1U |

#### 5.4.2.2.29. SECOC_VERIFICATIONFAILURE

| | |
|---|---|
| **Purpose** | Verification not successful. |
| **Value** | 1U |

#### 5.4.2.2.30. SECOC_VERIFICATIONSUCCESS

| | |
|---|---|
| **Purpose** | Verification successful. |
| **Value** | 0U |

#### 5.4.2.2.31. SECOC_VERIFICATION_STATUS_PROP_AUTOSAR

| | |
|---|---|
| **Purpose** | SecOC propagates the verification status via the AUTOSAR defined API(s). |
| **Value** | 1U |

#### 5.4.2.2.32. SECOC_VERIFICATION_STATUS_PROP_EB

| | |
|---|---|
| **Purpose** | SecOC propagates the verification status via the custom API(s). |
| **Value** | 2U |

#### 5.4.2.2.33. SECOC_VERIFICATION_STATUS_PROP_NONE

| | |
|---|---|
| **Purpose** | SecOC does not propagate the verification status. |
| **Value** | 0U |

### 5.4.2.3. Functions

#### 5.4.2.3.1. SecOCFreshnessValueFuncNameRx

| | |
|---|---|
| **Purpose** | This interface is used by the SecOC to obtain the current freshness value if SecOCUseAuthDataFreshness is disabled. The function name is configurable. |
| **Synopsis** | `Std_ReturnType` **`SecOCFreshnessValueFuncNameRx`** `( uint16 SecOCFreshnessValueID , const uint8 * SecOCTruncatedFreshnessValue , uint32 SecOCTruncatedFreshnessValueLength , uint16 SecOC-` |

| | | |
|---|---|---|
| | `CounterSyncAttempts , uint8 * SecOCFreshnessValue , uint32 * SecOCFreshnessValueLength );` | |
| **Parameters (in)** | `SecOCFreshnessValueID` | Holds the identifier of the freshness value. |
| | `SecOCTruncatedFreshnessValue` | Holds the truncated freshness value that was contained in the Secured I-PDU. |
| | `SecOCTruncatedFreshnessValue-Length` | Holds the length in bits of the truncated freshness value. |
| | `SecOCCounterSyncAttempts` | Holds the number of authentication verify attempts of this PDU since the last reception. The value is 0 for the first attempt and incremented on every unsuccessful verification attempt. |
| | `SecOCFreshnessValueLength` | Holds the length in bits of the freshness value. |
| **Parameters (out)** | `SecOCFreshnessValue` | Holds the freshness value to be used for the calculation of the authenticator. |
| **Return Value** | whether the request was successful or not. | |
| | `E_OK` | Request successful. |
| | `E_NOT_OK` | Request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. |
| | `E_BUSY` | The freshness information can temporarily not be provided. |

#### 5.4.2.3.2. SecOCFreshnessValueFuncNameRx_UseAuthDataFreshness

| | | |
|---|---|---|
| **Purpose** | This interface is used by the SecOC to obtain the current freshness value if SecOCUseAuthDataFreshness is enabled. The function name is configurable. | |
| **Synopsis** | `Std_ReturnType` **`SecOCFreshnessValueFuncNameRx_UseAuthDataFreshness`** `( uint16 SecOCFreshnessValueID , const uint8 * SecOCTruncatedFreshnessValue , uint32 SecOCTruncatedFreshnessValueLength , const uint8 * SecOCAuthDataFreshnessValue , uint16 SecOCAuthDataFreshnessValueLength , uint16 SecOCAuthVerifyAttempts , uint8 * SecOCFreshnessValue , uint32 * SecOCFreshnessValue-Length );` | |
| **Parameters (in)** | `SecOCFreshnessValueID` | Holds the identifier of the freshness value. |
| | `SecOCTruncatedFreshnessValue` | Holds the truncated freshness value that was contained in the Secured I-PDU. |

| | | |
|---|---|---|
| | SecOCTruncatedFreshnessValue-Length | Holds the length in bits of the truncated freshness value. |
| | SecOCAuthDataFreshnessValue | The parameter holds a part of the received, not yet authenticated PDU. |
| | SecOCAuthDataFreshnessValue-Length | This is the length value in bits that holds the freshness from the authentic PDU. |
| | SecOCAuthVerifyAttempts | Holds the number of authentication verify attempts of this PDU since the last reception. The value is 0 for the first attempt and incremented on every unsuccessful verification attempt. |
| | SecOCFreshnessValueLength | Holds the length in bits of the freshness value. |
| **Parameters (out)** | SecOCFreshnessValue | Holds the freshness value to be used for the calculation of the authenticator. |
| **Return Value** | whether the request was successful or not. | |
| | E_OK | Request successful. |
| | E_NOT_OK | Request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. |
| | E_BUSY | The freshness information can temporarily not be provided. |

#### 5.4.2.3.3. SecOCFreshnessValueFuncNameTx

| | | |
|---|---|---|
| **Purpose** | This interface is used by the SecOC to obtain the current freshness value if SecOCProvideTxTruncatedFreshnessValue is disabled. The function name is configurable. | |
| **Synopsis** | `Std_ReturnType` **`SecOCFreshnessValueFuncNameTx`** `( uint16 SecOCFreshnessValueID , uint8 * SecOCFreshnessValue , uint32 * SecOCFreshnessValueLength );` | |
| **Parameters (in)** | SecOCFreshnessValueID | Holds the identifier of the freshness value. |
| | SecOCFreshnessValueLength | Holds the length of the required freshness value in bits. |
| **Parameters (out)** | SecOCFreshnessValue | Holds the current freshness value. |
| **Return Value** | whether the request was successful or not. | |
| | E_OK | Request successful. |

| | | |
|---|---|---|
| | E_NOT_OK | Request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. |
| | E_BUSY | The freshness information can temporarily not be provided. |

#### 5.4.2.3.4. SecOCFreshnessValueFuncNameTx_TruncatedFreshnessValue

| | | |
|---|---|---|
| **Purpose** | This interface is used by the SecOC to obtain the current freshness value if SecOCProvideTxTruncatedFreshnessValue is enabled. The function name is configurable. | |
| **Synopsis** | `Std_ReturnType` **`SecOCFreshnessValueFuncNameTx_TruncatedFreshnessValue`** `( uint16 SecOCFreshnessValueID , uint8 * SecOCFreshnessValue , uint32 * SecOCFreshnessValueLength , uint8 * SecOCTruncatedFreshnessValue , uint32 * SecOCTruncatedFreshnessValueLength );` | |
| **Parameters (in)** | SecOCFreshnessValueID | Holds the identifier of the freshness value. |
| | SecOCFreshnessValueLength | Holds the length of the required freshness value in bits. |
| | SecOCTruncatedFreshnessValueLength | Holds the length of the required truncated freshness value in bits. |
| **Parameters (out)** | SecOCFreshnessValue | Holds the current freshness value. |
| | SecOCTruncatedFreshnessValue | Holds the current truncated freshness value. |
| **Return Value** | whether the request was successful or not. | |
| | E_OK | Request successful. |
| | E_NOT_OK | Request failed, a freshness value cannot be provided due to general issues for freshness or this FreshnessValueId. |
| | E_BUSY | The freshness information can temporarily not be provided. |

#### 5.4.2.3.5. SecOCMacGenerateStatusCallout

| | |
|---|---|
| **Purpose** | Configurable function which is called by the SecOC to report if the MAC could not be generated. The function name is configurable. |

| | | |
|---|---|---|
| **Synopsis** | `void` **`SecOCMacGenerateStatusCallout`** `( SecOC_MacGenerateStatusType macGenerateStatus );` | |
| **Parameters (in)** | `macGenerateStatus` | |

#### 5.4.2.3.6. SecOCRxShapeFuncName

| | | |
|---|---|---|
| **Purpose** | This interface is used by the SecOC to remove the padding within the secured PDU. The function name is configurable. | |
| **Synopsis** | `void` **`SecOCRxShapeFuncName`** `( PduIdType SecOCPduID , uint8 * SecPdu , const PduLengthType * SrcSecPduLength , PduLengthType * DstSecPduLength , uint32 AuthenticatorLength );` | |
| **Parameters (in)** | `SecOCPduID` | Holds the identifier of the secured PDU or the identifier of the received authentic PDU, when the Secured PDU Collection is used, at SecOC. |
| | `SrcSecPduLength` | Holds the length of the received secured PDU. |
| | `AuthenticatorLength` | Holds the length of the authenticator. |
| **Parameters (in,out)** | `SecPdu` | Holds the secured PDU. |
| | `DstSecPduLength` | in: Holds the maximum length of the secured PDU. out:Holds the length of the secured PDU without the padding. |

#### 5.4.2.3.7. SecOCSecuredPDUTransmittedFuncName

| | | |
|---|---|---|
| **Purpose** | This interface is used by the SecOC to indicate that the Secured I-PDU has been initiated for transmission. The function name is configurable. | |
| **Synopsis** | `void` **`SecOCSecuredPDUTransmittedFuncName`** `( uint16 SecOCFreshnessValueID );` | |
| **Parameters (in)** | `SecOCFreshnessValueID` | Holds the identifier of the freshness value. |

#### 5.4.2.3.8. SecOCTxShapeFuncName

| | |
|---|---|
| **Purpose** | This interface is used by the SecOC to to add the required padding within the secured PDU to maintain a fixed layout. The function name is configurable. |

| | | |
|---|---|---|
| **Synopsis** | `void` **`SecOCTxShapeFuncName`** `( PduIdType SecOCPduID , uint8 * SecPdu , const PduLengthType * SrcSecPduLength , PduLengthType * DstSecPduLength , uint32 AuthenticatorLength );` | |
| **Parameters (in)** | `SecOCPduID` | Holds the identifier of the received authentic PDU at SecOC. |
| | `SrcSecPduLength` | Holds the length of the generated secured PDU without the required padding. |
| | `AuthenticatorLength` | Holds the length of the authenticator. |
| **Parameters (in,out)** | `SecPdu` | Holds the secured PDU. |
| | `DstSecPduLength` | in: Holds the maximum length of the secured PDU. out: Holds the length of the secured PDU with the padding. |

#### 5.4.2.3.9. SecOCVerificationStatusCallout

| | | |
|---|---|---|
| **Purpose** | Configurable function which is called by the SecOC to report the verification attempt success The function name is configurable. | |
| **Synopsis** | `void` **`SecOCVerificationStatusCallout`** `( SecOC_VerificationStatusType verificationStatus );` | |
| **Parameters (in)** | `verificationStatus` | |

#### 5.4.2.3.10. SecOC_CancelTransmit

| | | |
|---|---|---|
| **Purpose** | Function to request the cancellation of an authentication and transmission of an Authentic I-PDU. If the Csm is used to authenticate the I-PDU, then the cancellation may take several main function cycles because the authentication sequence cannot be canceled at the CSM. | |
| **Synopsis** | `Std_ReturnType` **`SecOC_CancelTransmit`** `( PduIdType id );` | |
| **Parameters (in)** | `id` | ID of the Authentic I-PDU to be transmitted. |
| **Return Value** | the status of the cancellation request | |
| | `E_OK` | Cancellation request was performed successfully by the SecOC module. |
| | `E_NOT_OK` | Cancellation request was rejected. |

#### 5.4.2.3.11. SecOC_CopyTxData

| | | |
|---|---|---|
| **Purpose** | This function is called to acquire the transmit data of an I-PDU segment (N-PDU) for a Secured I-PDU. | |
| **Synopsis** | `BufReq_ReturnType` **`SecOC_CopyTxData`** `( PduIdType id , PduInfoType * info , RetryInfoType * retry , PduLengthType * availableDataPtr );` | |
| **Parameters (in)** | `id` | ID of the secured I-PDU to be transmitted. |
| | `info` | A pointer to a structure with Secured I-PDU related data that shall be transmitted: data length and pointer to I-SDU buffer |
| | `retry` | This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter shall point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data shall remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position. |
| **Parameters (out)** | `availableDataPtr` | Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs. |
| **Return Value** | the status of the request | |

| | | |
|---|---|---|
| | BUFREQ_OK | Data has been copied to the transmit buffer completely as requested |
| | BUFREQ_E_BUSY | Request could not be fulfilled, because the required amount of Tx data is not available. The LoTp module can either retry the request with the same PduInfoPtr or treat the return value like BUFREQ_E_NOT_OK. |
| | BUFREQ_E_NOT_OK | Data has not been copied. Request failed. |
| **Description** | This function is called to acquire the transmit data of an I-PDU segment (N-PDU) for a Secured I-PDU. Each call to this function provides the next part of the Secured I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr. | |

#### 5.4.2.3.12. SecOC_DeInit

| | | |
|---|---|---|
| **Purpose** | DeInit Function. | |
| **Synopsis** | `void SecOC_DeInit ( void );` | |
| **Description** | This service stops the secure onboard communication. All I-PDU buffers are cleared and have to be obtained again, if needed, after SecOC_Init has been called. By a call to SecOC_DeInit the AUTOSAR SecOC module is put into an not initialized state. | |

#### 5.4.2.3.13. SecOC_Init

| | | |
|---|---|---|
| **Purpose** | Init Function. | |
| **Synopsis** | `void SecOC_Init ( const SecOC_ConfigType * config );` | |
| **Parameters (in)** | config | Pointer to a selected configuration structure. |
| **Description** | This function initializes the SecOC module. | |

#### 5.4.2.3.14. SecOC_IsValidConfig

| | |
|---|---|
| **Purpose** | Validates the post-build configuration data structure. |

| | | |
|---|---|---|
| **Synopsis** | `Std_ReturnType` **`SecOC_IsValidConfig`** `( const void * voidConfigPtr );` | |
| **Parameters (in)** | `voidConfigPtr` | pointer to a SecOC post-build data structure. If a NULL_PTR is passed, the SecOC will attempt to retrieve the SecOC post-build configuration from the PbcfgM module. |
| **Return Value** | the status of the request | |
| | `E_OK` | When the pre-compile, link-time and platform hash values stored within the post-build structure correspond to the hash values of the compiled source files. |
| | `E_NOT_OK` | Otherwise, E_NOT_OK will be returned. |
| **Description** | This function validates the post-build configuration data structure passed to the SecOC_Init function. | |

#### 5.4.2.3.15. SecOC_MainFunctionRx

| | |
|---|---|
| **Purpose** | This function performs the processing of the SecOC module's authentication and verification processing for the Rx path. |
| **Synopsis** | `void` **`SecOC_MainFunctionRx`** `( void );` |

#### 5.4.2.3.16. SecOC_MainFunctionTx

| | |
|---|---|
| **Purpose** | This function performs the processing of the SecOC module's authentication and verification processing for the Tx path. |
| **Synopsis** | `void` **`SecOC_MainFunctionTx`** `( void );` |

#### 5.4.2.3.17. SecOC_RxIndication

| | | |
|---|---|---|
| **Purpose** | Service to indicate direct reception of a Secured I-PDU from a lower layer communication interface. | |
| **Synopsis** | `void` **`SecOC_RxIndication`** `( PduIdType id , PduInfoType * info );` | |
| **Parameters (in)** | `id` | ID of the received Secured I-PDU. |

| | | |
|---|---|---|
| | info | A pointer to a structure with Secured I-PDU related data that is received: data length and pointer to I-SDU buffer |
| **Description** | This call triggers the verification of the received Secured I-PDU. Called by the PduR. | |

#### 5.4.2.3.18. SecOC_TpTxConfirmation

| | | |
|---|---|---|
| **Purpose** | Service to confirm transmission via TP. | |
| **Synopsis** | `void SecOC_TpTxConfirmation ( PduIdType id , Std_ReturnType result );` | |
| **Parameters (in)** | id | ID of the transmitted Secured I-PDU. |
| | result | Result of transmission. |
| **Description** | The lower layer transport protocol module confirms the transmission of a Secured I-PDU via PduR. | |

#### 5.4.2.3.19. SecOC_Transmit

| | | |
|---|---|---|
| **Purpose** | Function to request authentication and transmission of an authentic I-PDU. | |
| **Synopsis** | `Std_ReturnType SecOC_Transmit ( PduIdType id , const PduInfoType * info );` | |
| **Parameters (in)** | id | ID of the Authentic I-PDU to be transmitted. |
| | info | A pointer to a structure with Authentic I-PDU related data that shall be transmitted: data length and pointer to I-SDU. |
| **Return Value** | whether the request was successful or not. | |
| | E_OK | Request successful. |
| | E_NOT_OK | Request failed. |

#### 5.4.2.3.20. SecOC_TriggerTransmit

| | |
|---|---|
| **Purpose** | Service to copy the Secured I-PDU to the lower layer. |

| | | |
|---|---|---|
| **Synopsis** | `Std_ReturnType` **`SecOC_TriggerTransmit`** `( PduIdType TxPduId , PduInfoType * PduInfoPtr );` | |
| **Parameters (in)** | `TxPduId` | ID of the SDU that is requested to be transmitted. |
| **Parameters (in,out)** | `PduInfoPtr` | A pointer to a buffer (SduDataPtr) to where the SDU data shall be copied and the available buffer size in SduLength. |
| **Return Value** | the result of the data copy process | |
| | `E_OK` | SDU has been copied and SduLength indicates the number of copied bytes. |
| | `E_NOT_OK` | No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data. |

#### 5.4.2.3.21. SecOC_TxConfirmation

| | | |
|---|---|---|
| **Purpose** | Service to confirm transmission. | |
| **Synopsis** | `void` **`SecOC_TxConfirmation`** `( PduIdType id );` | |
| **Parameters (in)** | `id` | ID of the transmitted Secured I-PDU. |
| **Description** | The lower layer communication interface module confirms the transmission of a Secured I-PDU via PduR. | |

#### 5.4.2.3.22. SecOC_VerifyStatusOverride

| | | |
|---|---|---|
| **Purpose** | This service enables the user to set the override verification status of a an I-PDU and to skip the verification procedure. | |
| **Synopsis** | `Std_ReturnType` **`SecOC_VerifyStatusOverride`** `( uint16 freshnessValueId , uint8 overrideStatus , uint8 numberOfMessagesToOverride );` | |
| **Parameters (in)** | `freshnessValueId` | Identifier of a specific Freshness Value |
| | `overrideStatus` | 0 = Override VerifyStatus to 'Fail' until further notice; 1 = Override VerifyStatus to 'Fail' until NumberOfMessagesToOverride is reached 2 = Cancel Override of VerifyStatus 41 = Override VerifyStatus to "Pass" until NumberOfMessagesToOverride is |

| | | |
|---|---|---|
| | | reached; only available if SecOCEnableForcedPassOverride is set to TRUE 43 = The verification procedure is skipped until further notice; only available if SecOCEnableForcedPassOverride is set to TRUE |
| | numberOfMessagesToOverride | Number of sequential VerifyStatus to override when using a specific counter for authentication verification. This is only considered when OverrideStatus is equal to 1 |
| **Return Value** | the status of the request | |
| | E_OK | request successful |
| | E_NOT_OK | request failed |
| **Description** | This Service provides the ability to override the VerifyStatus with 'Fail'/'Pass' or to skip the verification when using a specific Freshness Value to verify authenticity of data making up an I-PDU. Using this interface, VerifyStatus may be overridden 1. Indefinitely for received I-PDUs which use the specific Freshness Value for authentication verification 2. For a number of sequentially received I-PDUs which use the specific Freshness Value for authentication verification. 3. To skip the verification procedure for received I-PDUs which use the specific Freshness Value for authentication verification | |

## 5.4.3. Integration notes

### 5.4.3.1. Exclusive areas

This section describes the exclusive areas used by the SecOC module.

#### 5.4.3.1.1. SCHM_SECOC_EXCLUSIVE_AREA_0

| | |
|---|---|
| **Protected data structures** | This exclusive area protects the data structure SecOC_RxData[<PduId>] |
| **Recommended locking mechanism** | The locking mechanism for this exclusive area can be disabled if the following functions do not interrupt each other:<br>▶ SecOC_StartOfReception()<br>SecOC_RxIndication() |

| | |
|---|---|
| | ▶ SecOC_MainFunctionRx()<br><br>If the conditions listed above do not apply, the exclusive area shall be protected by a locking mechanism. The options for locking are described in the EB tresos AutoCore Generic documentation. Refer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details. |

#### 5.4.3.1.2. SCHM_SECOC_EXCLUSIVE_AREA_1

| | |
|---|---|
| **Protected data structures** | This exclusive area protects the data structures SecOC_TxData[<PduId>].TxBufferUsed. |
| **Recommended locking mechanism** | The locking mechanism for this exclusive area can be disabled if the following functions do not interrupt each other:<br><br>▶ SecOC_Transmit()<br>▶ SecOC_CancelTransmit()<br>▶ SecOC_TxConfirmation()<br>▶ SecOC_TpTxConfirmation()<br>▶ SecOC_MainFunctionTx()<br><br>If the conditions listed above do not apply, the exclusive area shall be protected by a locking mechanism. The options for locking are described in the EB tresos AutoCore Generic documentation. Refer to the section Mapping exclusive areas in the basic software modules in the Integration notes section for details. |

### 5.4.3.2. Production errors

Production errors are not reported by the SecOC module.

### 5.4.3.3. Memory mapping

General information about memory mapping is provided in the EB tresos AutoCore Generic documentation. Refer to the section Memory mapping and compiler abstraction in the Integration notes section for details.

The following table provides the list of sections that may be mapped for this module:

| Memory section |
|---|
| CODE |
| VAR_CLEARED_BOOLEAN |
| VAR_CLEARED_UNSPECIFIED |
| VAR_INIT_UNSPECIFIED |
| VAR_INIT_8 |
| VAR_CLEARED_8 |
| CONST_32 |
| CONST_UNSPECIFIED |
| CONFIG_DATA_UNSPECIFIED |

### 5.4.3.4. Integration requirements

**WARNING**

**Integration requirements list is not exhaustive**

The following list of integration requirements helps you to integrate your product. However, this list is not exhaustive. You also require information from the user's guide, release notes, and EB tresos AutoCore known issues to successfully integrate your product.

#### 5.4.3.4.1. SecOC.Req.Integration_MacUniformProcType

| | |
|---|---|
| **Description** | All Csm MacGenerate or MacVerify jobs referenced by the SecOC module for I-PDU authentication and verification need to be either synchronous or asynchronous. |
| **Rationale** | The current implementation of the SecOC module only offers a global configuration parameter to select between Csm synchronous or asynchronous job processing types. |

#### 5.4.3.4.2. SecOC.Req.Integration_Init

| | |
|---|---|
| **Description** | *SecOC_Init()* initializes the module. *SecOC_Init()* shall be called during the start-up procedure of the ECU before any other API of the module is called. It is allowed to call the *SecOC_MainFunctionRx()* or *SecOC_MainFunctionTx()* before the initialization. |

#### 5.4.3.4.3. SecOC.Req.Integration_DeInit

| | |
|---|---|
| **Description** | The function *SecOC_DeInit()* deinitializes the module. *SecOC_DeInit()* shall be called during the shutdown procedure of the ECU. |

#### 5.4.3.4.4. SecOC.Req.Integration_MainFuncRxCycleTime

| | |
|---|---|
| **Description** | The *SecOC_MainFunctionRx()* shall be called with a sufficient cycle time depending on the received data. Example: If the fastest I-PDU in the lower layer is transmitted with a cycle time of 10 ms, the *SecOC_MainFunctionRx()* needs to be called with the same or a lower cycle time.<br><br>Note: If Csm is used synchronously as the provider of cryptographic functionality, the cryptographic calculations are executed directly within the *SecOC_MainFunctionRx()* context. Therefore, the run-time of the *SecOC_MainFunctionRx()* might be significantly higher than if you use a Csm module asynchronously. The overall time consumption for verification is lower when synchronous job processing is used. |

#### 5.4.3.4.5. SecOC.Req.Integration_RxScheduledNetworks

| | |
|---|---|
| **Description** | For scheduled networks like FlexRay, the *SecOC_MainFunctionRx()* shall be scheduled to synchronize to the network. |
| **Rationale** | This avoids authentication failures caused by the discontinuity of the freshness value. |

#### 5.4.3.4.6. SecOC.Req.Integration_MainFuncTxCycleTime

| | |
|---|---|
| **Description** | The *SecOC_MainFunctionTx()* shall be called with a sufficient cycle time depending on the transmitted data. Example: If the fastest I-PDU in the lower layer is transmitted with a cycle time of 10 ms, the *SecOC_MainFunctionTx()* needs to be called with the same or a lower cycle time.<br><br>Note: If Csm is used synchronously as the provider of cryptographic functionality, the cryptographic calculations are executed directly within the *SecOC_MainFunctionTx()* context. Therefore, the run-time of the *SecOC_MainFunctionTx()* might be significantly higher than if you use the Csm module asynchronously. The overall time consumption for message authentication is lower when synchronous job processing is used. |

#### 5.4.3.4.7. SecOC.Req.Integration_TxScheduledNetworks

| | |
|---|---|
| **Description** | For scheduled networks like FlexRay, the *SecOC_MainFunctionTx()* shall be scheduled to synchronize to the network. |
| **Rationale** | This avoids authentication failures caused by the discontinuity of the freshness value. |

#### 5.4.3.4.8. SecOC.Req.Integration_PropagateVerificationStatus

| | |
|---|---|
| **Description** | To propagate the verification status via CFUNC or RTE, *SecOCPropagateVerificationStatus* must set to a value different that *NONE*. NOTE: In order to have Autosar compliant interfaces to propagate the verification status, the option *AUTOSAR* must be set. |

# 6. Bibliography

# Bibliography

[1] *AUTOSAR Specification of Crypto Service Manager*, Issue 4.3.0, Publisher: AUTOSAR

[2] *AUTOSAR Specification of Crypto Interface*, Issue 4.3.0, Publisher: AUTOSAR

[3] *AUTOSAR Specification of Module Secure Onboard Communication*, Issue 4.3.0, Publisher: AUTOSAR

[4] *AUTOSAR Specification of Crypto Driver*, Issue 4.3.0, Publisher: AUTOSAR