MICROSAR Classic CryIf

Technical Reference

Crypto Interface

Version 7.1.0

Authors	vismss, visrpp, vismaw, vismxe, rstemmler, kwiedom, coechsler
Status	Released

Document Information

History

Author	Date	Version	Remarks
vismss	2017-03-07	1.01.00	Initial creation of Technical Reference
visrpp	2017-05-08	1.02.00	Changed chapter 4.1.6, 4.1.7, 4.1.8, 4.1.12
vismaw	2019-03-15	4.01.00	KeyElementCopyPartial, APIs added
vismxe	2019-12-11	5.00.00	Chapter 2.1: Support for asynchronous key handling Chapter 5.2: add chapter
rstemmler	2023-03-03	6.00.00	Product name updated to MICROSAR Classic Chapter 2.1: Support for asynchronous key handling Chapter 2.1.1: Autosar deviations Chapter 2.5.1: add new service IDs Chapter 3.1.1: add new file Chapter 3.1.2: add new file Chapter 4.1: add new services Chapter 5.2: update to new template
kwiedom	2023-05-04	7.00.00	Remove init state handling for multi core Chapter 2.1.1: add as Autosar deviation Chapter 2.2: change initialization description Chapter 2.5.1: remove Service IDs and Error Codes Chapters 4.1.1, 4.1.2: update description
coechsler	2023-03-11	7.01.00	Added chapter 6 Cybersecurity Updated Table 7-2

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_CryptoInterface.pdf	4.3.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	4.3.0
[3]	AUTOSAR	AUTOSAR_SWS_CryptoInterface.pdf	4.4.0
[4]	AUTOSAR	AUTOSAR_SWS_CryptoInterface.pdf	R20-11
[5]	AUTOSAR	AUTOSAR_SWS_CryptoInterface.pdf	R21-11

Contents

Illustrations

Tables

1 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CRYIF as specified in [1], [3], [4], [5].

Supported Configuration Variants:	pre-compile	
Vendor ID:	CRYIF_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CRYIF_MODULE_ID	112 decimal (according to [1])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The Crypto Interface (CRYIF) is called by the Cryptographic Service Manager (CSM) to forward its service requests to the underlying Crypto Drivers (CRYPTO). The CRYIF has access to the CRYPTO to calculate results with their cryptographic services. These results are returned to the CSM by the CRYIF.

1.1 Architecture Overview

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

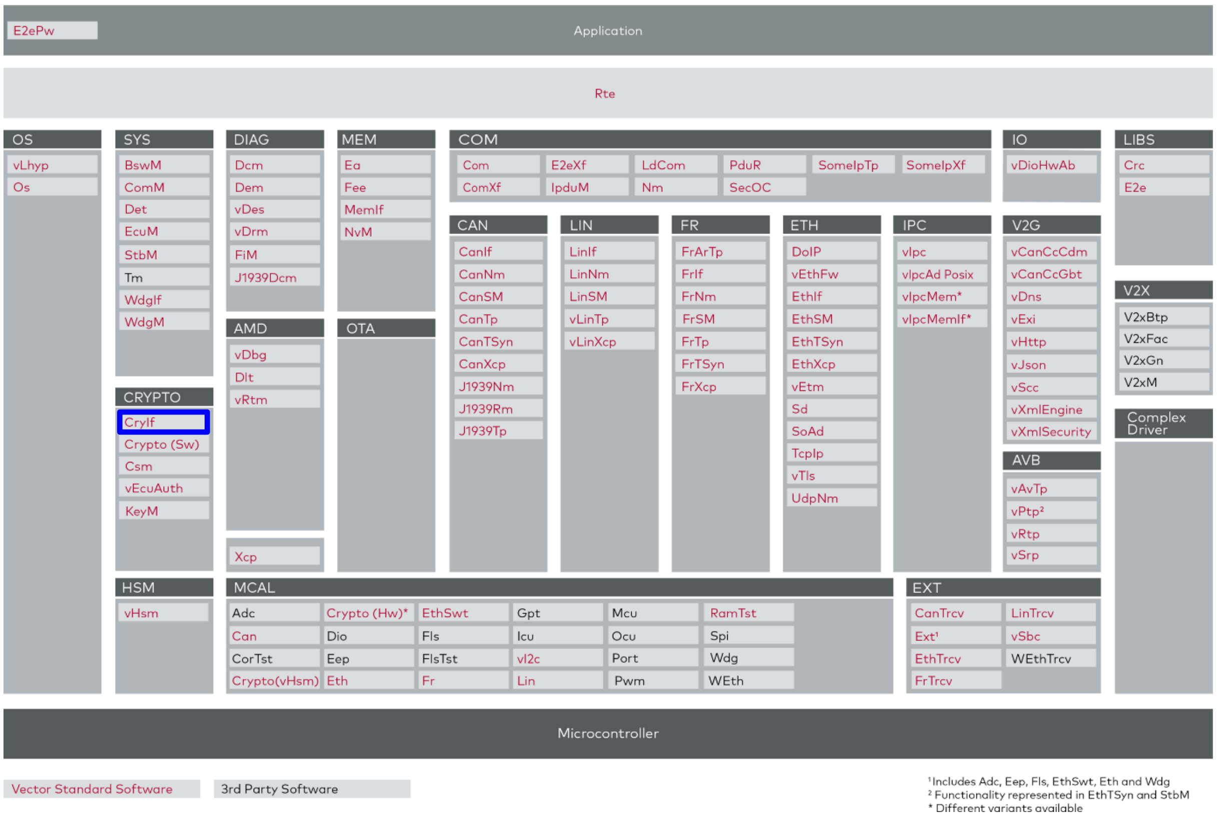

Figure 1-1 AUTOSAR Architecture Overview

The next figure shows the interfaces to adjacent modules of the CRYIF. These interfaces are described in chapter 4.

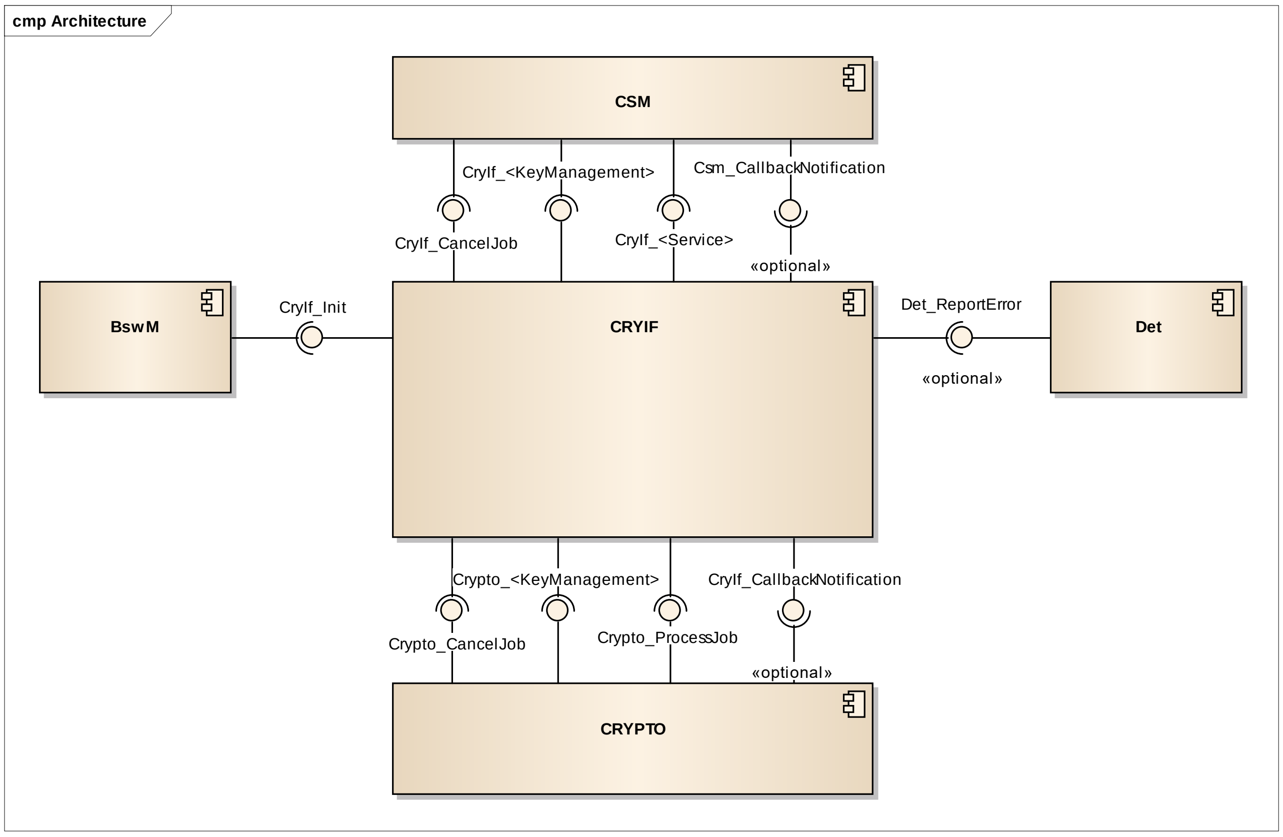

Figure 1-2 Interfaces to adjacent modules of the CRYIF

2 Functional Description

2.1 Features

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

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

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Dispatching jobs to the configured Crypto Driver
Dispatching key management functionalities

Table 2-1 Supported AUTOSAR standard conform features according to [1]

The following features specified in [3] are supported:

Supported AUTOSAR Standard Conform Features
Redirection of input and output buffers to key elements
KeyElementCopyPartial API
Processing of asynchronous key management jobs
Forwarding of Callback Notifications

Table 2-2 Supported AUTOSAR standard conform features according to [3]

The following features specified in [4] are supported:

Supported AUTOSAR Standard Conform Features
KeyGetStatus API
KeySetInvalid API

Table 2-3 Supported AUTOSAR standard conform features according to [4]

2.1.1 Deviations

The following features are not supported in a strictly standard-compliant way:

Features with deviations against the AUTOSAR Standard
Forwarding of Callback Notifications according to AUTOSAR 4.3.1 and earlier ([1])
Parameter check keyLength != 0 (SWS_CryIf_00053) in CryIf_KeyElementSet according to AUTOSAR R21-11 ([5]) and earlier ([4], [3], [1]).
Initialization of CryIf module is not tracked, and no related DETs are reported (SWS_CryIf_00009 and e.g., SWS_CryIf_00014)

Table 2-4 Features with deviations against the AUTOSAR Standard

Forwarding of Callback Notifications is not supported by CryIf in a way that strictly complies with [1]. According to AUTOSAR 4.3.x, the function `CryIf_CallbackNotification` receives an argument `job` as a const pointer. The CryIf should pass this argument to the CSM as a var pointer. This is an inconsistency in the AUTOSAR specification that has been fixed in AUTOSAR 4.4.0. The MICROSAR Classic CryIf does not support the faulty interface defined in AUTOSAR 4.3.x and instead uses a var pointer for the argument `job` (see chapter 4.3.1).

Parameter check keyLength != 0 (SWS_CryIf_00053) in CryIf_KeyElementSet is specified since [1]. The check is not executed to be able to support key deletion requests.

2.1.2 KeyElementCopyPartial

If the underlying Crypto driver does not support the Crypto_KeyElementCopyPartial API, the CryIf handles partial copying by itself. The copying of the CryIf also takes place if the cryIfKeyId and the targetCryIfKeyId point to different Crypto drivers.

First, the procedure gets the contents of both key elements via Crypto_KeyElementGet API. Afterwards, the partial copying takes place by copying the requested part of the source key element to the requested location in the target key element. Last, the constructed key element is written to the Crypto driver via Crypto_KeyElementSet API.

If the target offset is larger than the length of the target key element, a zero-padding is applied in between.

2.2 Initialization

The CRYIF has no RAM variables, as such `CryIf_Init()` and `CryIf_InitMemory()` do nothing. No development error detection or reporting related to the initialization state is performed.

2.3 States

The CRYIF does not have a state machine.

2.4 Main Functions

CRYIF does not provide a main function. All calls are synchronous.

2.5 Error Handling

2.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `CRYIF_DEV_ERROR_REPORT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator but it must have the same signature as the service `Det_ReportError()`.

The reported CRYIF ID is 112.

The reported service IDs identify the services which are described in 4.1. The following table presents the service IDs and the related services:

Service ID	Service
0x01	CryIf_GetVersionInfo
0x02	CryIf_ProcessJob
0x03	CryIf_CancelJob
0x04	CryIf_KeyElementSet
0x05	CryIf_KeySetValid
0x06	CryIf_KeyElementGet
0x0f	CryIf_KeyElementCopy
0x10	CryIf_KeyCopy
0x07	CryIf_RandomSeed
0x08	CryIf_KeyGenerate
0x09	CryIf_KeyDerive
0x0A	CryIf_KeyExchangeCalcPubVal
0x0B	CryIf_KeyExchangeCalcSecret
0x0C	CryIf_CertificateParse
0x11	CryIf_CertificateVerify
0x12	CryIf_KeyElementCopyPartial
0x13	CryIf_KeyGetStatus
0x14	CryIf_KeySetInvalid

Table 2-5 Service IDs

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

Error Code	Description
0x02	API request called with invalid parameter (null pointer)
0x03	API request called with invalid parameter (out of range)
0x04	API request called with invalid parameter (invalid value)
0x12	API request called but request is not supported by Crypto

Table 2-6 Errors reported to DET

3 Integration

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

3.1 Scope of Delivery

The delivery of the CRYIF contains the files which are described in the chapters 3.1.1 and 3.1.2:

3.1.1 Static Files

File Name	Description
CryIf.c	This file contains the CRYIF source code.
CryIf.h	This is the header file of the CRYIF.
CryIf_Cbk.h	This is the callback header file of CRYIF.
CryIf_Private.h	This is the header file for stub functions. These functions are used internally if a service is not configured in CryIf / not supported by Crypto.

Table 3-1 Static files

3.1.2 Dynamic Files

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

File Name	Description
CryIf_Cfg.c	This is configuration source file.
CryIf_Cfg.h	This is configuration header file.
CryIf_MemMap.h	Memory section mapping for CryIf. In deliveries of MICROSAR Classic R27 and newer, this file is generated. In older releases it is static.

Table 3-2 Generated files

4 API Description

For an interfaces overview please see Figure 1-2.

4.1 Services provided by CRYIF

4.1.1 CryIf_InitMemory

Prototype	
`void CryIf_InitMemory (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
No functionality since no global memory is used.	
Particularities and Limitations	
Function provided to meet API requirements.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 4-1 CryIf_InitMemory

4.1.2 CryIf_Init

Prototype	
`void CryIf_Init (void)`	
Parameter	
ConfigPtr [in]	Configuration structure for initializing the module
Return code	
void	none
Functional Description	
No functionality since no variables are used and initialization state is not tracked.	
Particularities and Limitations	
Function provided to meet API requirements.	
Call context	
> TASK > This function is Synchronous	

> This function is Non-Reentrant

Table 4-2 CryIf_Init

4.1.3 CryIf_GetVersionInfo

Prototype	
`void CryIf_GetVersionInfo (Std_VersionInfoType *versioninfo)`	
Parameter	
versioninfo [out]	Pointer to where to store the version information. Parameter must not be NULL.
Return code	
void	none
Functional Description	
Returns the version information.	
Particularities and Limitations	
none CryIf_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component.	
Call context	
> TASK|ISR2 > This function is Synchronous > This function is Reentrant	

Table 4-3 CryIf_GetVersionInfo

4.1.4 CryIf_ProcessJob

Prototype	
`Std_ReturnType CryIf_ProcessJob (uint32 channelId, Crypto_JobType *job)`	
Parameter	
channelId [in]	Holds the identifier of the crypto channel.
job [in,out]	Pointer to the configuration of the job. Contains structures with user and primitive relevant information.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_NOT_VALID Request failed, the key is not valid.
	CRYPTO_E_QUEUE_FULL Request failed, the queue is full.

	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Process the received job.	
Particularities and Limitations	
none This interface dispatches the received jobs to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-4 CryIf_ProcessJob

4.1.5 CryIf_CancelJob

Prototype	
`Std_ReturnType **CryIf_CancelJob** (uint32 channelId, Crypto_JobType *job)`	
Parameter	
channelId [in]	Holds the identifier of the crypto channel.
job [in,out]	Pointer to the configuration of the job. Contains structures with user and primitive relevant information.
Return code	
Std_ReturnType	E_OK Request successful, job has been removed.
Std_ReturnType	E_NOT_OK Request failed, job could not be removed.
Functional Description	
Cancels the received job.	
Particularities and Limitations	
none This interface removes the provided job from the underlying Crypto Driver Object queue.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-5 CryIf_CancelJob

4.1.6 CryIf_KeyElementSet

Prototype	
`Std_ReturnType` **`CryIf_KeyElementSet`** `(uint32 cryIfKeyId, uint32 keyElementId, const uint8 *keyPtr, uint32 keyLength)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key whose key element shall be set.
keyElementId [in]	Holds the identifier of the key element which shall be set.
keyPtr [in]	Holds the pointer to the key data which shall be set as key element.
keyLength [in]	Contains the length of the key element in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element size does not match size of provided data.
Functional Description	
Sets a key element.	
Particularities and Limitations	
This function shall dispatch the key element set function to the configured crypto driver object. NULL_PTR check for P2CONST keyPtr is not executed to save runtime. Zero value check for keyLength is not executed to be able to support key deletion.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-6 CryIf_KeyElementSet

4.1.7 CryIf_KeySetValid

Prototype	
`Std_ReturnType` **`CryIf_KeySetValid`** `(uint32 cryIfKeyId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key whose key elements shall be set to valid.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.

	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Sets the key to valid.	
Particularities and Limitations	
none This function shall dispatch the key set valid function to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-7 CryIf_KeySetValid

4.1.8 CryIf_KeyElementGet

Prototype	
`Std_ReturnType` **`CryIf_KeyElementGet`** `(uint32 cryIfKeyId, uint32 keyElementId, uint8 *resultPtr, uint32 *resultLengthPtr)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key whose key element shall be set.
keyElementId [in]	Holds the identifier of the key element which shall be set.
resultPtr [in]	Holds the pointer to the result data which shall be set as key element.
resultLengthPtr [in]	Contains the length of the result in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Exports the key element	
Particularities and Limitations	
none This function shall dispatch the get key element function to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-8 CryIf_KeyElementGet

4.1.9 CryIf_KeyElementCopy

Prototype	
`Std_ReturnType` **`CryIf_KeyElementCopy`** `(uint32 cryIfKeyId, uint32 keyElementId, uint32 targetCryIfKeyId, uint32 targetKeyElementId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key whose key element shall be the source element.
keyElementId [in]	Holds the identifier of the key element which shall be the source for the copy operation.
targetCryIfKeyId [in]	Holds the identifier of the key whose key element shall be the destination element.

targetKeyElementId [in]	Holds the identifier of the key element which shall be the destination for the copy operation.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_EXTRACT_DENIED Request failed, not allowed to extract key material.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element sizes are not compatible.
Functional Description	
Copy key element.	
Particularities and Limitations	
none This function shall copy a key element from one key to a target key.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-9 CryIf_KeyElementCopy

4.1.10 CryIf_KeyElementCopyPartial

Prototype	
`Std_ReturnType` **`CryIf_KeyElementCopyPartial`** `(uint32 cryIfKeyId, uint32 keyElementId, uint32 keyElementSourceOffset, uint32 keyElementTargetOffset, uint32 keyElementCopyLength, uint32 targetCryIfKeyId, uint32 targetKeyElementId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key whose key element shall be the source element.
keyElementId [in]	Holds the identifier of the key element which shall be the source for the copy operation.
keyElementSourceOffset [in]	This is the offset of the source key element indicating the start index of the copy operation.
keyElementTargetOffset [in]	This is the offset of the destination key element indicating the start index of the copy operation.
keyElementCopyLength [in]	Specifies the number of bytes that shall be copied.
targetCryIfKeyId [in]	Holds the identifier of the key whose key element shall be the destination element.

targetKeyElementId [in]	Holds the identifier of the key element which shall be the destination for the copy operation.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element sizes are not compatible.
	CRYPTO_E_KEY_EMPTY Request failed because of uninitialized source key element.
Functional Description	
Copies a key element partially.	
Particularities and Limitations	
none This function copies a key element partially from one key to a target key.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-10 CryIf_KeyElementCopyPartial

4.1.11 CryIf_KeyCopy

Prototype	
`Std_ReturnType` **`CryIf_KeyCopy`** `(uint32 cryIfKeyId, uint32 targetCryIfKeyId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key whose key element shall be the source element.
targetCryIfKeyId [in]	Holds the identifier of the key whose key element shall be the destination element.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_KEY_READ_FAIL Request failed, read access was denied.
	CRYPTO_E_KEY_WRITE_FAIL Request failed, write access was denied.
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed, the key is not available.
	CRYPTO_E_KEY_SIZE_MISMATCH Request failed, the key element sizes are not compatible.
Functional Description	
Copy the key.	
Particularities and Limitations	
none This function shall copy all key elements from the source key to a target key.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-11 CryIf_KeyCopy

4.1.12 CryIf_RandomSeed

Prototype	
`Std_ReturnType` **`CryIf_RandomSeed`** `(uint32 cryIfKeyId, const uint8 *seedPtr, uint32 seedLength)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key for which a new material shall be generated.
seedPtr [in]	Holds a pointer to the memory location which contains the data to feed the seed.
seedLength [in]	Contains the length of the seed in bytes.
Return code	
Std_ReturnType	E_OK Request successful.

	E_NOT_OK Request failed.
Functional Description	
Initialize the seed.	
Particularities and Limitations	
none This function shall dispatch the random seed function to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-12 CryIf_RandomSeed

4.1.13 CryIf_KeyGenerate

Prototype	
`Std_ReturnType **CryIf_KeyGenerate** (uint32 cryIfKeyId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key which is to be updated with the generated value.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Generates a key.	
Particularities and Limitations	
none This function shall dispatch the key generate function to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-13 CryIf_KeyGenerate

4.1.14 CryIf_KeyDerive

Prototype
`Std_ReturnType **CryIf_KeyDerive** (uint32 cryIfKeyId, uint32 targetCryIfKeyId)`

Parameter	
cryIfKeyId [in]	Holds the identifier of the key which is used for key derivation.
targetCryIfKeyId [in]	Holds the identifier of the key which is used to store the derived key.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Derives a key.	
Particularities and Limitations	
none This function shall dispatch the key derive function to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-14 CryIf_KeyDerive

4.1.15 CryIf_KeyExchangeCalcPubVal

Prototype	
`Std_ReturnType` **`CryIf_KeyExchangeCalcPubVal`** `(uint32 cryIfKeyId, uint8 *publicValuePtr, uint32 *publicValueLengthPtr)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key which shall be used for the key exchange protocol.
publicValuePtr [out]	Contains the pointer to the data where the public value shall be stored.
publicValueLengthPtr [in,out]	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.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Calculation of the public value.	

Particularities and Limitations
none This function shall dispatch the key exchange public value calculation function to the configured crypto driver object.
Call context
> TASK > This function is Synchronous > This function is Reentrant

Table 4-15 CryIf_KeyExchangeCalcPubVal

4.1.16 CryIf_KeyExchangeCalcSecret

Prototype	
`Std_ReturnType` **`CryIf_KeyExchangeCalcSecret`** `(uint32 cryIfKeyId, const uint8 *partnerPublicValuePtr, uint32 partnerPublicValueLength)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key which shall be used for the key exchange protocol.
partnerPublicValuePtr [in]	Holds the pointer to the memory location which contains the partner's public value.
partnerPublicValueLength [in]	Contains the length of the partner's public value in bytes.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
	CRYPTO_E_SMALL_BUFFER Request failed, the provided buffer is too small to store the result.
Functional Description	
Calculation of the secret.	
Particularities and Limitations	
none This function shall dispatch the key exchange common shared secret calculation function to the configured crypto driver object.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-16 CryIf_KeyExchangeCalcSecret

4.1.17 CryIf_CertificateParse

Prototype	
`Std_ReturnType` **`CryIf_CertificateParse`** `(uint32 cryIfKeyId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key slot in which the certificate has been stored.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Parse stored certificate.	
Particularities and Limitations	
none This function shall dispatch the certificate parse function to the configured crypto driver object. If API support is disabled (/MICROSAR/CryIf/CryIfCryptoModule/CryIfSupportsCertificateAPI) for a specific driver, the service will always return E_NOT_OK for this driver.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-17 CryIf_CertificateParse

4.1.18 CryIf_CertificateVerify

Prototype	
`Std_ReturnType` **`CryIf_CertificateVerify`** `(uint32 cryIfKeyId, uint32 verifyCryIfKeyId, Crypto_VerifyResultType *verifyPtr)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key which shall be used to validate the certificate.
verifyCryIfKeyId [in]	Holds the identifier of the key containing the certificate, which shall be verified.
verifyPtr [out]	Holds a pointer to the memory location which will contain the result of the certificate verification.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Certificate verification.	

Particularities and Limitations
none Verifies the certificate stored in the key referenced by verifyCryptoKeyId with the certificate stored in the key referenced by cryIfKeyId. If API support is disabled (/MICROSAR/CryIf/CryIfCryptoModule/CryIfSupportsCertificateAPI) for a specific driver, the service will always return E_NOT_OK for this driver.
Call context
> TASK > This function is Synchronous > This function is Reentrant

Table 4-18 CryIf_CertificateVerify

4.1.19 CryIf_KeySetInvalid

Prototype	
`Std_ReturnType **CryIf_KeySetInvalid** (uint32 cryIfKeyId)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key which shall be set to invalid.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Set crypto key to invalid.	
Particularities and Limitations	
none This function shall dispatch the key set invalid function to the configured crypto driver object. If API support is disabled (/MICROSAR/CryIf/CryIfCryptoModule/CryIfSupportsKeyStatusAPI) for a specific driver, the service will always return E_NOT_OK for this driver.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-19 CryIf_KeySetInvalid

4.1.20 CryIf_KeyGetStatus

Prototype	
`Std_ReturnType, CRYIF_CODE) **CryIf_KeyGetStatus** (uint32 cryIfKeyId, P2VAR(Crypto_KeyStatusType, AUTOMATIC, CRYIF_APPL_VAR) keyStatusPtr)`	
Parameter	
cryIfKeyId [in]	Holds the identifier of the key which shall be set to invalid.

keyStatusPtr[out]	Holds a pointer to the memory where the key status shall be written to.
Return code	
Std_ReturnType	E_OK Request successful.
	E_NOT_OK Request failed.
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy.
Functional Description	
Get status of a key.	
Particularities and Limitations	
none This function shall dispatch the key get status function to the configured crypto driver object. If API support is disabled (/MICROSAR/CryIf/CryIfCryptoModule/CryIfSupportsKeyStatusAPI) for a specific driver, the service will always return E_NOT_OK for this driver.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 4-20 CryIf_KeyGetStatus

4.2 Services used by CRYIF

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

Component	API
DET	Det_ReportError

Table 4-21 Services used by the CRYIF

4.3 Callback Functions

This chapter describes the callback function that is implemented by the CRYIF and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `CryIf_Cbk.h` by the CRYIF.

4.3.1 CryIf_CallbackNotification

Prototype	
`void` **`CryIf_CallbackNotification`** `( Crypto_JobType *job, Std_ReturnType result )`	
Parameter	
`job`	Points to the completed job's information structure. It contains a callbackID to identify which job is finished.
`result`	Contains the result of the cryptographic operation.

Return code	
`void`	none
Functional Description	
Notifies the CRYIF about the completion of the request with the result of the cryptographic operation.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant.	

Table 4-22 CryIf_CallbackNotification

5 Configuration

5.1 Configuration Variants

The CRYIF supports the configuration variants

- `VARIANT-PRE-COMPILE`

The configuration classes of the CRYIF parameters depend on the supported configuration variants. For their definitions please see the CryIf_bswmd.arxml file.

5.2 Configurable Attributes

Since CRYIF has some variants in its behavior, a configuration is necessary. To ease up, CRYIF offers a configuration description file (CryIf_bswmd.arxml). Most of the settings have reasonable default values and do not need to be adapted. The description of each configurable option is described within its help in the Configurator 5 tool.

6 Cybersecurity

This chapter describes relevant information for a secure integration and configuration, so-called Cybersecurity Manual Items (CMI), of this component to fulfill identified Technical Cybersecurity Requirements (TCR). Additionally, functional cybersecurity dependencies to other components are described.

6.1 Configuration

For this component, there are no special measures to be taken in the configuration regarding cybersecurity.

6.2 Runtime Interfaces: BSW

TCR	Depends on Component(s)	Comment
TCR-40, TCR-41, TCR-60, TCR-61	CRYPTO	Needed for all cryptographic services

Table 6-1

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
CSM	Crypto Service Manager
CRYIF	Crypto Interface
CRYPTO	Crypto Driver

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
CMI	Cybersecurity Manual Item
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPORT	Provide Port
RPORT	Require Port
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
TCR	Technical Cybersecurity Requirement

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com