Document Title	Specification of Crypto Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	806
Document Classification	Standard

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.0

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2016-11-30	4.3.0	AUTOSAR Release Management	Initial Release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software Module Crypto Interface (CRYIF).

The Crypto Interface module is located between the low level Crypto solutions (Crypto Driver [4] and SW-based CDD) and the upper service layer (Crypto Service Manager [5]). It represents the interface to the services of the Crypto Driver(s) for the upper service layer. A AUTOSAR Layered View can be found in Figure 7.1.

The Crypto Interface module provides a unique interface to manage different Crypto HW and SW solutions like HSM, SHE or SW-based CDD. Thus multiple underlying internal and external Crypto HW as well as SW solutions can be utilized by the Crypto Service Manager module based on a mapping scheme maintained by Crypto Interface.

2 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to the Crypto Interface module that are not included in the AUTOSAR glossary [7].

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

2.1 Glossary of Terms

Terms:	***Description:***	
Crypto Driver Object	A Crypto Driver Object is an instance of a crypto module (hardware or software), which is able to perform one or more different crypto operations.	
Key	A Key can be referenced by a job in the Csm. In the Crypto Driver, the key references a specific key type.	
Key Type	A key type consists of references to key elements. The key types are typically pre-configured by the vendor of the Crypto Driver.	
Key Element	Key elements are used to store data. This data can be e.g. key material or the IV needed for AES encryption. It can also be used to configure the behaviour oft he key management functions.	
Channel	A channel is the path from a Crypto Service Manager queue via the Crypto Interface to a specific Crypto Driver Object.	
Job	A job is an instance of a configured cryptographic primitive.	
Crypto Primitive	A crypto primitive is an instance of a configured cryptographic algorithm.	
Operation	An operation of a crypto primitive declares what part of the crypto primitive shall be performed. There are three different operations:	
	START	Operation indicates a new request of a crypto primitive, and it shall cancel all previous requests.
	UPDATE	Operation indicates, that the crypto primitive expect input data.
	FINISH	Operation indicates, that after this part all data are fed completely and the crypto primitive can finalize the calculations.
	It is also possible to perform more than one operation at once by	

	concatenating the corresponding bits of the operation mode argument.	
Priority	The priority of a job defines the importance of it. The higher the priority (as well in value), the more immediate the job will be executed. The priority of a cryptographic job is part of the configuration.	
Processing	Indicates the kind of job processing.	
	Asynchronous	The job is not processed immediately when calling a corresponding function. Usually, the caller is informed via a callback function when the job has been finished.
	Synchronous	The job is processed immediately when calling a corresponding function. When the function returns, a result will be available.

3 Related documentation

3.1 Input documents

[1] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[3] AUTOSAR General Specification for Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[4] AUTOSAR Specification of Crypto Driver
AUTOSAR_SWS_CryptoDriver.pdf

[5] AUTOSAR Specification of Crypto Service Manager
AUTOSAR_SWS_CryptoServiceManager.pdf

[6] AUTOSAR Requirements on Crypto Modules
AUTOSAR_SRS_CryptoStack.pdf

[7] Glossary
AUTOSAR_TR_Glossary

3.2 Related standards and norms

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

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General) [3], which is also valid for Crypto Interface.

Thus, the specification SWS BSW General [3] shall be considered as additional and required specification for Crypto Interface.

4 Constraints and assumptions

4.1 Limitations

The Crypto Interface is specifically designed to operate with one or multiple underlying Crypto Drivers. Several Crypto Driver modules covering different HW processing units or cores are represented by just one generic interface as specified in the Crypto Driver specification [4].
Any software based Crypto Driver shall be implemented as a CDD represented by the same interface above.

4.2 Applicability to car domains

The Crypto Interface can be used for all domain applications when security features are to be used.

5 Dependencies to other modules

[SWS_CryIf_00001] ⌈The Crypto Interface (CRYIF) shall be able to be called by the Crypto Service Manager (CSM), and forward its service requests to the underlying Crypto Drivers.
⌋()

[SWS_CryIf_00002] ⌈The CRYIF shall be able to access the underlying Crypto Drivers to calculate results with their cryptographic services. These results shall be returned back to the CSM by the CRYIF.
⌋()

5.1 File structure

5.1.1 Code file structure

[SWS_CryIf_00003] ⌈ The code file structure shall not be defined within this specification completely.
⌋()

[SWS_CryIf_00004] ⌈ The code file structure shall contain one source file CryIf.c, that contains the entire CRYIF code.
⌋()

5.1.2 Header file structure

[SWS_CryIf_00005] ⌈The header file structure shall contain an application interface header file CryIf.h, which provides the function prototypes to access the CRYIF services.
⌋()

[SWS_CryIf_00006] ⌈The header file structure shall contain a callback interface CryIf_Cbk.h, which provides the callback function prototype to be used by the underlying crypto drivers.
⌋(SRS_BSW_00346)

[SWS_CryIf_00007] ⌈ The header file structure shall contain a configuration header CryIf_Cfg.h, that provides the configuration parameters for the CRYIF.
⌋(SRS_BSW_00345)

[SWS_CryIf_00008] ⌈The Figure (CRYIF file dependencies) shows the include file structure, which shall be as follows:
CryIf.h shall include Csm_Types.h and CryIf_Cfg.h.
CryIf.c shall include CryIf.h, Crypto.h and Csm_Cbk.h.
CryIf_Cbk.h shall include CryIf.h.
⌋(SRS_BSW_00348)

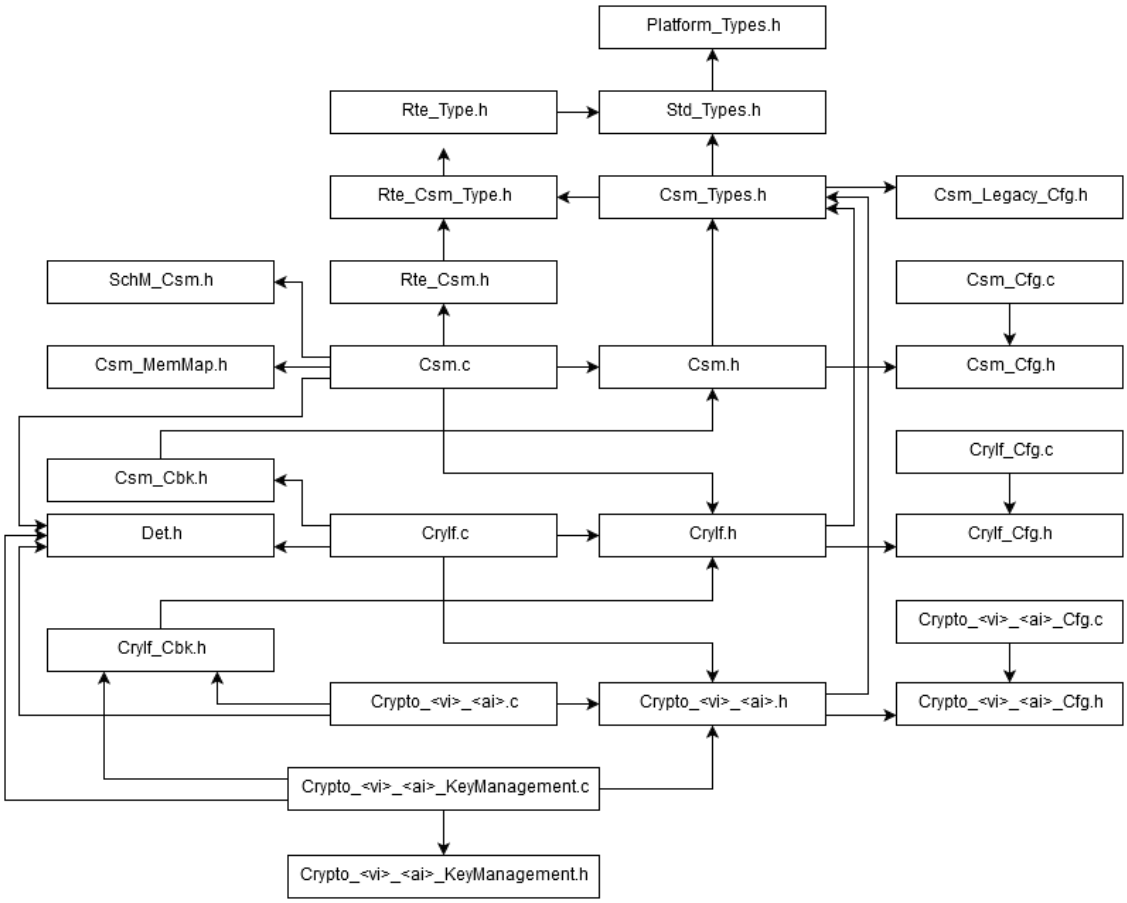


Figure 5.1: CRYIF file dependencies

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_CryIf_91000
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_CryIf_00007
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_CryIf_00006
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_CryIf_00008
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_CryIf_91000
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_CryIf_91013
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_CryIf_91013
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_CryIf_91001
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_CryIf_91000
SRS_CryptoStack_00034	The Crypto Interface shall report detected development errors to the Default Error Tracer	SWS_CryIf_00014, SWS_CryIf_00016, SWS_CryIf_00017, SWS_CryIf_00027, SWS_CryIf_00028, SWS_CryIf_00029, SWS_CryIf_00049, SWS_CryIf_00050, SWS_CryIf_00052, SWS_CryIf_00053, SWS_CryIf_00056, SWS_CryIf_00057, SWS_CryIf_00059, SWS_CryIf_00060, SWS_CryIf_00062, SWS_CryIf_00063,

		SWS_CryIf_00064, SWS_CryIf_00068, SWS_CryIf_00069, SWS_CryIf_00070, SWS_CryIf_00071, SWS_CryIf_00073, SWS_CryIf_00074, SWS_CryIf_00076, SWS_CryIf_00077, SWS_CryIf_00082, SWS_CryIf_00083, SWS_CryIf_00084, SWS_CryIf_00085, SWS_CryIf_00086, SWS_CryIf_00090, SWS_CryIf_00091, SWS_CryIf_00092, SWS_CryIf_00093, SWS_CryIf_00094, SWS_CryIf_00098, SWS_CryIf_00099, SWS_CryIf_00104, SWS_CryIf_00107, SWS_CryIf_00108, SWS_CryIf_00110, SWS_CryIf_00111, SWS_CryIf_00112, SWS_CryIf_00113, SWS_CryIf_00115, SWS_CryIf_00116, SWS_CryIf_00117, SWS_CryIf_00118, SWS_CryIf_00119, SWS_CryIf_00121, SWS_CryIf_00122, SWS_CryIf_00123, SWS_CryIf_00124, SWS_CryIf_00125, SWS_CryIf_00126, SWS_CryIf_00127, SWS_CryIf_00129, SWS_CryIf_00130, SWS_CryIf_00131
SRS_CryptoStack_00086	The CSM module shall distinguish between error types	SWS_CryIf_00009

7 Functional specification

The Crypto Interface is located between the Crypto Service Manager and the underlying crypto drivers and is the unique interface to access cryptographic operations for all upper layers (BSW). The Crypto Interface is also the only user of the crypto drivers and provides a unique interface to manage different crypto hardware and software solutions. The Abstraction Layer encapsulates different mechanisms of hardware and software access, so the Crypto Interface implementation is independent from the underlying Crypto Drivers which can be realized in hardware or software.
Also it ensures the concurrent access to crypto services to enable the possibility to process multiple crypto tasks at the same time.

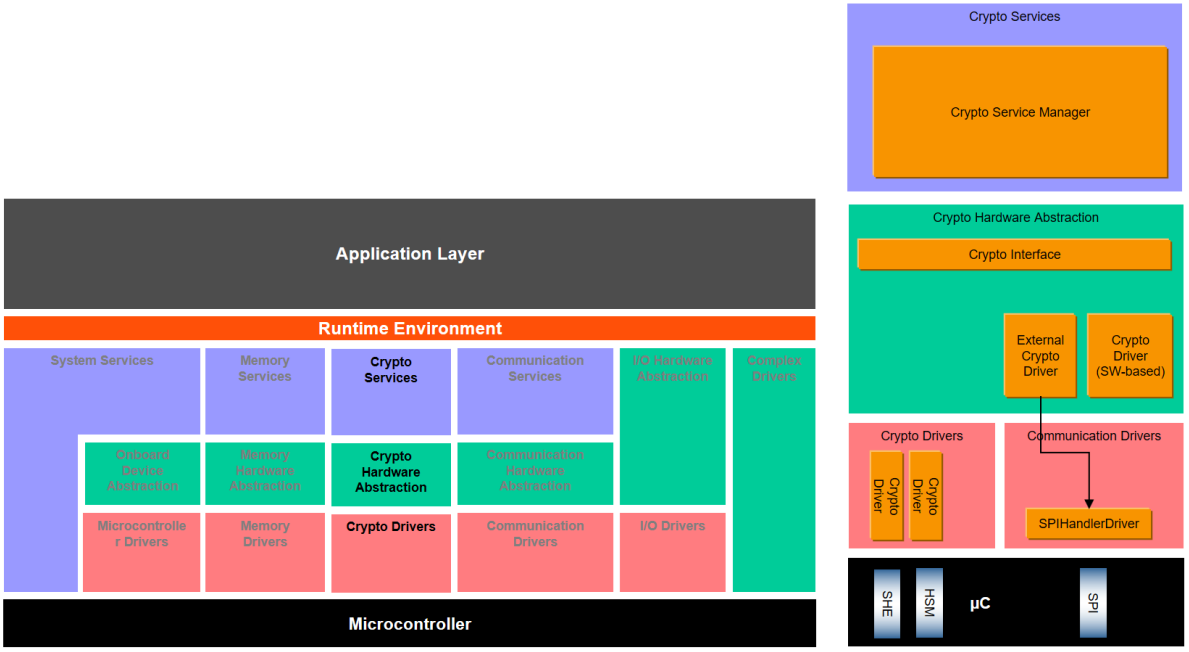


Figure 7.1: AUTOSAR Layered View with Crypto-Interface

7.1 Error classification

7.1.1 Development Errors

[SWS_CryIf_00009] Development Error Types[

Type of error	*Related error code*	*Value [hex]*
API request called before initialisation of CRYIF module.	CRYIF_E_UNINIT	0x00
Initialisation of CRYIF module failed.	CRYIF_E_INIT_FAILED	0x01
API request called with invalid parameter (null	CRYIF_E_PARAM_POINTER	0x02

pointer).		
API request called with invalid parameter (out of range).	CRYIF_E_PARAM_HANDLE	0x03
API request called with invalid parameter (invalid value).	CRYIF_E_PARAM_VALUE	0x04

] (SRS_CryptoStack_00086)

7.1.2 Runtime Errors

There are no runtime errors.

7.1.3 Transient Faults

There are no transient faults.

7.1.4 Production Errors

There are no production errors.

7.1.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

In this chapter, all types included from the following files are listed:

[SWS_CryIf_00011] [Imported Types

Module	*Imported Type*
Csm	Crypto_JobType
	Crypto_VerifyResultType
	Csm_ResultType
Std_Types	Std_ReturnType
	Std_VersionInfoType

]()

The Crypto Stack API uses the following extension to Std_ReturnType:
[SWS_CryIf_00012] [

Range:	CRYPTO_E_BUSY	0x02	The service request failed because the service is still busy
	CRYPTO_E_SMALL_BUFFER	0x03	The service request failed because the provided buffer is too small to store the result
	CRYPTO_E_ENTROPY_EXHAUSTION	0x04	The service request failed because the entropy of the random number generator is exhausted
	CRYPTO_E_QUEUE_FULL	0x05	The service request failed because the queue is full
	CRYPTO_E_KEY_READ_FAIL	0x06	</br></tr>
	CRYPTO_E_KEY_WRITE_FAIL	0x07	The service request failed because the writing access failed
	CRYPTO_E_KEY_NOT_AVAILABLE	0x08	The service request failed because the key is not available
	CRYPTO_E_KEY_NOT_VALID	0x09	The service request failed because the key is invalid.
	CRYPTO_E_KEY_SIZE_MISMATCH	0x0A	The service request failed because the key size does not match.
	CRYPTO_E_COUNTER_OVERFLOW	0x0B	The service request failed because the counter is overflowed.
	CRYPTO_E_JOB_CANCELED	0x0C	The service request failed because the Job has been canceled.
Description:	--		

] ()
The Crypto Stack API uses the key element index definition from the CSM module.

8.2 Type Definitions

There are no type definitions.

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 General API

8.3.1.1 CryIf_Init

[SWS_CryIf_91000] [

Service name:	CryIf_Init	

Syntax:	`void CryIf_Init(`
	`  void`
	`)`	

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

] (SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)

[SWS_CryIf_00014] [If the initialization of the CRYIF module fails, the CRYIF shall report `CRYIF_E_INIT_FAILED` to the DET.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00015] [The service CryIf_Init() shall initialize the global variables and data structures of the CRYIF including flags and buffers.
] ( )

8.3.1.2 CryIf_GetVersionInfo

[SWS_CryIf_91001] [

Service name:	CryIf_GetVersionInfo	

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

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	versioninfo	Pointer to where to store the version information of this module.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	Returns the version information of this module.	

] (SRS_BSW_00407)

[SWS_CryIf_00016] [If default error detection for the CRYIF module is enabled: The function `CryIf_GetVersionInfo` shall raise the error `CRYIF_E_UNINIT` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00017] [If default error detection for the CRYIF module is enabled: The function `CryIf_GetVersionInfo` shall raise the error `CRYIF_E_PARAM_POINTER` if the parameter `versioninfo` is a null pointer.
] (SRS_CryptoStack_00034)

8.3.2 Job Processing Interface

8.3.2.1 CryIf_ProcessJob

To unite a single call function and a streaming approach for the crypto services, there is one interface `CryIf_ProcessJob()`. Its `Crypto_JobType job` parameter contains a `Crypto_OperationModeType` flag field (`job->jobPrimitiveInputOutput.mode`), which can be set as "`START`", "`UPDATE`", "`FINISH`" or combination of them. It declares explicitly what operation shall be performed. These operation modes can be mixed, and execute multiple operations at once.

To process a crypto service with a single call with `Crypto_ProcessJob()` the operation mode is a disjunction of the 3 modes "`START|UPDATE|FINISH`".

[SWS_CryIf_91003] [

Service name:	CryIf_ProcessJob	
Syntax:	`Std_ReturnType CryIf_ProcessJob( uint32 channelId, Crypto_JobType* job )`	
Service ID[hex]:	0x03	
Sync/Async:	Sync or Async, depends on the configuration	
Reentrancy:	Reentrant	
Parameters (in):	channelId	Holds the identifier of the crypto channel.
Parameters (inout):	job	Pointer to the configuration of the job. Contains structures with user and primitive relevant information.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed CRYPTO_E_BUSY: Request Failed, Crypro Driver Object is Busy CRYPTO_E_KEY_NOT_VALID, Request failed, the key is not valid CRYPTO_E_KEY_SIZE_MISMATCH, Request failed, a key element has the wrong size. CRYIF_E_QUEUE_FULL: Request failed, the queue is full CRYIF_E_SMALL_BUFFER: The provided buffer is too small to store the result CRYPTO_E_JOB_CANCELED: The service request failed because the synchronous Job has been canceled.
Description:	This interface dispatches the received jobs to the configured crypto driver object.	

] ()

[SWS_CryIf_00027] [If default error detection for the CRYIF is enabled: The function `CryIf_ProcessJob` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00028] [If default error detection for the CRYIF is enabled: The function `CryIf_ProcessJob` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `channelId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00029] [If default error detection for the CRYIF is enabled: The function `CryIf_ProcessJob` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `job` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00044] [If no errors are detected by CRYIF, the service `CryIf_ProcessJob()` shall call `Crypto_<vi>_<ai>_ProcessJob()` for the driver configuration mapped to the service and pass on the return value.
]()

8.3.3 Job Cancellation Interface

8.3.3.1 CryIf_CancelJob

[SWS_CryIf_91014] [

Service name:	CryIf_CancelJob	
Syntax:	`Std_ReturnType CryIf_CancelJob( uint32 channelId, Crypto_JobType* job )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	channelId	Holds the identifier of the crypto channel.
Parameters (inout):	job	Pointer to the configuration of the job. Contains structures with user and primitive relevant information.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful, job has been removed E_NOT_OK: Request Failed, job couldn't be removed
Description:	This interface dispatches the job cancellation function to the configured crypto driver object.	

] ()

[SWS_CryIf_00129] [If default error detection for the CRYIF is enabled: The function `CryIf_CancelJob` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00130] [If default error detection for the CRYIF is enabled: The function `CryIf_CancelJob` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `channelId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00131] [If default error detection for the CRYIF is enabled: The function `CryIf_CancelJob` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `job` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00132] [If no errors are detected by CRYIF, the service `CryIf_CancelJob()` shall call `Crypto_<vi>_<ai>_CancelJob()` for the driver configuration mapped to the service and pass on the return value.
]()

8.3.4 Key Management Interface

8.3.4.1 Key Setting Interface

8.3.4.1.1 CryIf_KeyElementSet

[SWS_CryIf_91004] [

Service name:	CryIf_KeyElementSet	
Syntax:	`Std_ReturnType CryIf_KeyElementSet( uint32 cryIfKeyId, uint32 keyElementId, const uint8* keyPtr, uint32 keyLength )`	
Service ID[hex]:	0x04	
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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request failed CRYPTO_E_BUSY: Request failed, Crypto Driver Object is busy CRYPTO_E_KEY_WRITE_FAIL:Request failed because write access was denied CRYPTO_E_KEY_NOT_AVAILABLE: Request failed because the key is not available. CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element size does not match size of provided data.
Description:	This function shall dispatch the set key element function to the configured crypto driver object.	

] ()

[SWS_CryIf_00049] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementSet` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00050] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementSet` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00052] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementSet` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `keyPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00053] [If default error detection for the Crypto Driver is enabled: The function `CryIf_KeyElementSet` shall raise the error `CRYIF_E_PARAM_VALUE` and return `E_NOT_OK` if `keyLength` is zero.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00055] [If no errors are detected by CRYIF, the service `CryIf_KeyElementSet()` shall call `Crypto_<vi>_<ai>_KeyElementSet()` for the driver configuration mapped to the service and pass on the return value.
] ()

8.3.4.1.2 CryIf_KeySetValid

[SWS_CryIf_91005] [

Service name:	CryIf_KeySetValid	
Syntax:	`Std_ReturnType CryIf_KeySetValid( uint32 cryIfKeyId )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	cryIfKeyId	Holds the identifier of the key whose key elements shall be set to valid.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed CRYPTO_E_BUSY: Request Failed, Crypro Driver Object is Busy
Description:	This function shall dispatch the set key valid function to the configured crypto driver object.	

] ()

[SWS_CryIf_00056] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeySetValid` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00057] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeySetValid` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00058] [If no errors are detected by CRYIF, the service `CryIf_KeySetValid()` shall call `Crypto_<vi>_<ai>_KeySetValid()` for the driver configuration mapped to the service and pass on the return value.
] ()

8.3.4.2 Key Extraction Interface

8.3.4.2.1 CryIf_KeyElementGet

[SWS_CryIf_91006] [

Service name:	CryIf_KeyElementGet	
Syntax:	`Std_ReturnType CryIf_KeyElementGet( uint32 cryIfKeyId, uint32 keyElementId, uint8* resultPtr, uint32* resultLengthPtr )`	
Service ID[hex]:	0x06	
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 (inout):	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:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed CRYPTO_E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed because read access was denied CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result
Description:	This function shall dispatch the get key element function to the configured crypto driver object.	

] ()

[SWS_CryIf_00059] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementGet` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00060] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementGet` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00062] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementGet` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `resultPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00063] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementGet` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `resultLengthPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00064] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyElementGet` shall raise the error `CRYIF_E_PARAM_VALUE` and return `E_NOT_OK` if the value, which is pointed by `resultLengthPtr`, is zero.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00065] [If no errors are detected by CRYIF, the service `CryIf_KeyElementGet()` shall call `Crypto_<vi>_<ai>_KeyElementGet()` for the driver configuration mapped to the service and pass on the return value.
] ( )

8.3.4.3 Key Copying Interface

8.3.4.3.1 CryIf_KeyElementCopy

[SWS_CryIf_91015] [

Service name:	CryIf_KeyElementCopy	

Syntax:	`Std_ReturnType CryIf_KeyElementCopy(`
	`  uint32 cryIfKeyId,`
	`  uint32 keyElementId,`
	`  uint32 targetCryIfKeyId,`
	`  uint32 targetKeyElementId`
	`)`	

Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for the same cryIfKeyId	
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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed CRYPTO_E_BUSY: Request Failed, Crypto Driver Object is

		Busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available 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:	This function shall copy a key elements from one key to a target key.	

] ()

[SWS_CryIf_00110] [If default error detection for the CRYIF is enabled: The function `CryIf_KeyElementCopy` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00111] [If default error detection for the CRYIF is enabled: The function `CryIf_KeyElementCopy` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00112] [If default error detection for the CRYIF is enabled: The function `CryIf_KeyElementCopy` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `targetCryIfKeyId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00113] [If no errors are detected by CRYIF and the `cryIfKeyId` and `targetCryIfKeyId` are located in the same Crypto Driver, the service `CryIf_KeyElementCopy()` shall call `Crypto_<vi>_<ai>_KeyElementCopy()` for the driver configuration mapped to the service and pass on the return value.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00114] [If no errors are detected by CRYIF and the `cryIfKeyId` and `targetCryIfKeyId` are located in different Crypto Drivers, the service `CryIf_KeyElementCopy()` shall copy the provided key element by getting the element with `Crypto_<vi>_<ai>_KeyElementGet()` and setting the target key element via `Crypto_<vi>_<ai>_KeyElementSet()`.
] ( )

[SWS_CryIf_00115] [
If a key element of `cryIfKeyId` is not available in `targetCryIfKeyId`, the key element shall not be copied and no error code shall be returned.
If the source element size does not match the target key elements size, `CryIf_KeyElementCopy()` shall raise the error `CRYIF_E_KEY_SIZE_MISMATCH` and return `E_NOT_OK`.
] (SRS_CryptoStack_00034)

8.3.4.3.2 CryIf_KeyCopy

[SWS_CryIf_91016] [

Service name:	CryIf_KeyCopy	
Syntax:	`Std_ReturnType CryIf_KeyCopy( uint32 cryIfKeyId, uint32 targetCryIfKeyId )`	
Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for the same cryIfKeyId	
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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_KEY_NOT_AVAILABLE: Request failed, the requested key element is not available CRYPTO_E_KEY_READ_FAIL: Request failed, not allowed to extract key element CRYPTO_E_KEY_WRITE_FAIL: Request failed, not allowed to write key element. CRYPTO_E_KEY_SIZE_MISMATCH: Request failed, key element sizes are not compatible.
Description:	This function shall copy all key elements from the source key to a target key.	

] ()

[SWS_CryIf_00116] [If default error detection for the CRYIF is enabled: The function `CryIf_KeyCopy` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00117] [If default error detection for the CRYIF is enabled: The function `CryIf_KeyCopy` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00118] [If default error detection for the CRYIF is enabled: The function `CryIf_KeyCopy` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `targetCryIfKeyId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00119] [If no errors are detected by CRYIF and the `cryIfKeyId` and `targetCryIfKeyId` are located in the same Crypto Driver, the service `CryIf_KeyCopy()` shall call `Crypto_<vi>_<ai>_KeyElementCopy()` for the driver configuration mapped to the service and pass on the return value.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00120] [If no errors are detected by CRYIF and the `cryIfKeyId` and `targetCryIfKeyId` are located in different Crypto Drivers, the service

`CryIf_KeyCopy()` shall copy the provided key element by getting the element with `Crypto_<vi>_<ai>_KeyElementGet()` and setting the target key element via `Crypto_<vi>_<ai>_KeyElementSet()`.
] ( )

[SWS_CryIf_00121] [
If a key element of `cryIfKeyId` is not available in `targetCryIfKeyId`, the key element shall not be copied and no error code shall be returned.
If the source element size does not match the target key elements size, `CryIf_KeyCopy()` shall raise the error `CRYIF_E_KEY_SIZE_MISMATCH` and return `E_NOT_OK`.
] (SRS_CryptoStack_00034)

8.3.4.4 Key Generation Interface

8.3.4.4.1 CryIf_RandomSeed

[SWS_CryIf_91007] [

Service name:	CryIf_RandomSeed	
Syntax:	`Std_ReturnType CryIf_RandomSeed( uint32 cryIfKeyId, const uint8* seedPtr, uint32 seedLength )`	
Service ID[hex]:	0x07	
Sync/Async:	Sync or Async, depends on the configuration	
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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	This function shall dispatch the random seed function to the configured crypto driver object.	

] ()

[SWS_CryIf_00068] [If default error detection for the CRYIF is enabled: The function `CryIf_RandomSeed` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00069] [If default error detection for the CRYIF is enabled: The function `CryIf_RandomSeed` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00070] [If default error detection for the CRYIF is enabled: The function `CryIf_RandomSeed` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `seedPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00071] [If default error detection for the CRYIF is enabled: The function `CryIf_RandomSeed` shall raise the error `CRYIF_E_PARAM_VALUE` and return `E_NOT_OK` if `seedLength` is zero.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00072] [If no errors are detected by CRYIF, the service `CryIf_RandomSeed()` shall call `Crypto_<vi>_<ai>_RandomSeed()` for the driver configuration mapped to the service and pass on the return value.
] ( )

8.3.4.4.2 CryIf_KeyGenerate

[SWS_CryIf_91008] [

Service name:	CryIf_KeyGenerate	
Syntax:	`Std_ReturnType CryIf_KeyGenerate( uint32 cryIfKeyId )`	
Service ID[hex]:	0x08	
Sync/Async:	Sync or Async, depends on the configuration	
Reentrancy:	Reentrant	
Parameters (in):	cryIfKeyId	Holds the identifier of the key which is to be updated with the generated value.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy
Description:	This function shall dispatch the key generate function to the configured crypto driver object.	

] ()

[SWS_CryIf_00073] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyGenerate` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00074] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyGenerate` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out or range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00075] [If no errors are detected by CRYIF, the service `CryIf_KeyGenerate()` shall call `Crypto_<vi>_<ai>_KeyGenerate()` for the driver configuration mapped to the service and pass on the return value.
] ( )

8.3.4.5 Key Derivation Interface

8.3.4.5.1 CryIf_KeyDerive

[SWS_CryIf_91009] ⌈

Service name:	CryIf_KeyDerive	
Syntax:	`Std_ReturnType CryIf_KeyDerive( uint32 cryIfKeyId, uint32 targetCryIfKeyId )`	
Service ID[hex]:	0x09	
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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	This function shall dispatch the key derive function to the configured crypto driver object.	

⌋ ()

[SWS_CryIf_00076] ⌈ If default error detection for the CRYIF module is enabled: The function `CryIf_KeyDerive` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
⌋ (SRS_CryptoStack_00034)

[SWS_CryIf_00077] ⌈ If default error detection for the CRYIF module is enabled: The function `CryIf_KeyDerive` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out or range.
⌋ (SRS_CryptoStack_00034)

[SWS_CryIf_00122] ⌈ If default error detection for the CRYIF module is enabled: The function `CryIf_KeyDerive` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `targetCryIfKeyId` is out or range.
⌋ (SRS_CryptoStack_00034)

[SWS_CryIf_00081] ⌈ If no errors are detected by CRYIF, the service `CryIf_KeyDerive()` shall call `Crypto_<vi>_<ai>_KeyDerive()` for the driver configuration mapped to the service and pass on the return value.
⌋ ( )

The key derivation service needs a salt and password to derivate a new key. The salt and the password therefore are stored as key elements in the key referred by `cryIfKeyId`.

8.3.4.6 Key Exchange Interface

8.3.4.6.1 CryIf_KeyExchangeCalcPubVal

[SWS_CryIf_91010] ⌈

Service name:	CryIf_KeyExchangeCalcPubVal	
Syntax:	`Std_ReturnType CryIf_KeyExchangeCalcPubVal( uint32 cryIfKeyId, uint8* publicValuePtr, uint32* publicValueLengthPtr )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	cryIfKeyId	Holds the identifier of the key which shall be used for the key exchange protocol.
Parameters (inout):	publicValueLengthPtr	Holds a pointer to the memory location in which the public value length information is stored. On calling this function, this parameter shall contain the size of the buffer provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
Parameters (out):	publicValuePtr	Contains the pointer to the data where the public value shall be stored.
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result
Description:	This function shall dispatch the key exchange public value calculation function to the configured crypto driver object.	

] ()

[SWS_CryIf_00082] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcPubVal` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00083] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcPubVal` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00084] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcPubVal` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `publicValuePtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00085] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcPubVal` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `pubValueLengthPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00086] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcPubVal` shall raise the error

CRYIF_E_PARAM_VALUE and return E_NOT_OK if the value, which is pointed by pubValueLengthPtr, is zero.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00087] [If no errors are detected by CRYIF, the service CryIf_KeyExchangeCalcPubVal() shall call Crypto_<vi>_<ai>_KeyExchangeCalcPubVal() for the driver configuration mapped to the service and pass on the return value.
] ( )

8.3.4.6.2 CryIf_KeyExchangeCalcSecret

[SWS_CryIf_91011] [

Service name:	CryIf_KeyExchangeCalcSecret	
Syntax:	`Std_ReturnType CryIf_KeyExchangeCalcSecret( uint32 cryIfKeyId, const uint8* partnerPublicValuePtr, uint32 partnerPublicValueLength )`	
Service ID[hex]:	0x0b	
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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy CRYPTO_E_SMALL_BUFFER: The provided buffer is too small to store the result
Description:	This function shall dispatch the key exchange common shared secret calculation function to the configured crypto driver object.	

] ()

[SWS_CryIf_00090] [If default error detection for the CRYIF module is enabled: The function CryIf_KeyExchangeCalcSecret shall raise the error CRYIF_E_UNINIT and return E_NOT_OK if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00091] [If default error detection for the CRYIF module is enabled: The function CryIf_KeyExchangeCalcSecret shall raise the error CRYIF_E_PARAM_HANDLE and return E_NOT_OK if the parameter cryIfKeyId is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00092] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcSecret` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `partnerPublicValuePtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00093] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcSecret` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `partnerPubValueLengthPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00094] [If default error detection for the CRYIF module is enabled: The function `CryIf_KeyExchangeCalcSecret` shall raise the error `CRYPTO_E_PARAM_VALUE` and return `E_NOT_OK` if `partnerPubValueLength` is zero.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00095] [If no errors are detected by CRYIF, the service `CryIf_KeyExchangeCalcSecret()` shall call `Crypto_<vi>_<ai>_KeyExchangeCalcSecret()` for the driver configuration mapped to the service and pass on the return value.
] ( )

8.3.4.7 Certificate Interface

8.3.4.7.1 CryIf_CertificateParse

[SWS_CryIf_91012] [

Service name:	CryIf_CertificateParse	
Syntax:	`Std_ReturnType CryIf_CertificateParse( uint32 cryIfKeyId )`	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	cryIfKeyId	Holds the identifier of the key which shall be parsed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed E_BUSY: Request Failed, Crypto Driver Object is Busy
Description:	This function shall dispatch the certificate parse function to the configured crypto driver object.	

] ()

[SWS_CryIf_00098] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateParse` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00099] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateParse` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00104] [If no errors are detected by CRYIF, the service `CryIf_CertificateParse()` shall call `Crypto_<vi>_<ai>_CertificateParse()` for the driver configuration mapped to the service and pass on the return value.
] (SRS_CryptoStack_00034)

8.3.4.7.2 CryIf_CertificateVerify

[SWS_CryIf_91017] [

Service name:	CryIf_CertificateVerify	
Syntax:	`Std_ReturnType CryIf_CertificateVerify( uint32 cryIfKeyId, uint32 verifyCryIfKeyId, Crypto_VerifyResultType* verifyPtr )`	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant, but not for the same cryIfKeyId	
Parameters (in):	cryIfKeyId	Holds the identifier of the key which shall be used to validate the certificate.
	verifyCryIfKeyId	Holds the identifier of the key containing the certificate to be verified.
Parameters (inout):	None	
Parameters (out):	verifyPtr	Holds a pointer to the memory location which will contain the result of the certificate verification.
Return value:	Std_ReturnType	E_OK: Request successful E_NOT_OK: Request Failed
Description:	Verifies the certificate stored in the key referenced by verifyCryIfKeyId with the certificate stored in the key referenced by cryIfKeyId.	

] ()

[SWS_CryIf_00123] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateVerify` shall raise the error `CRYIF_E_UNINIT` and return `E_NOT_OK` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00124] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateVerify` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `cryIfKeyId` is out of range.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00125] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateVerify` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the parameter `validateCryIfKeyId` is out of range.

] (SRS_CryptoStack_00034)

[SWS_CryIf_00126] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateVerify` shall raise the error `CRYIF_E_PARAM_HANDLE` and return `E_NOT_OK` if the keys identified by `validateCryIfKeyId` and `cryIfKeyId` are not located in the same Crypto Driver.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00127] [If default error detection for the CRYIF module is enabled: The function `CryIf_CertificateVerify` shall raise the error `CRYIF_E_PARAM_POINTER` and return `E_NOT_OK` if the parameter `verifyPtr` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00128] [If no errors are detected by CRYIF, the service `CryIf_CertificateVerify()` shall call `Crypto_<vi>_<ai>_CertificateVerify()` for the driver configuration mapped to the service and pass on the return value.
] ( )

8.4 Call-back notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file CryIf_Cbk.h

8.4.1 CryIf_CallbackNotification

[SWS_CryIf_91013] [

Service name:	CryIf_CallbackNotification	
Syntax:	`void CryIf_CallbackNotification( const Crypto_JobType* job, Std_ReturnType result )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	void	--
Description:	Notifies the CRYIF about the completion of the request with the result of the cryptographic operation.	

] (SRS_BSW_00359, SRS_BSW_00360)

[SWS_CryIf_00107] [If default error detection for the CRYIF module is enabled: The function `CryIf_CallbackNotification` shall raise the error `CRYIF_E_UNINIT` if the module is not yet initialized.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00108] [If default error detection for the CRYIF module is enabled: The function `CryIf_CallbackNotification` shall raise the error `CRYIF_E_PARAM_POINTER` if the parameter `job` is a null pointer.
] (SRS_CryptoStack_00034)

[SWS_CryIf_00109] [If no errors are detected by CRYIF, the service `CryIf_CallbackNotification()` shall call `Csm_CallbackNotification()` and pass on the result.
] ( )

8.5 Expected Interfaces

8.5.1 Mandatory Interfaces

This chapter defines all interfaces, which are required to fulfill the core functionality of the CryIf module.

API function	*Description*
Csm_CallbackNotification	Notifies the CSM that a job has finished. This function is used by the underlying layer (CRYIF).

8.5.2 Optional Interfaces

This chapter defines all interfaces, which are required to fulfill an optional functionality of the CryIf module.

API function	*Description*

9 Sequence diagrams

N/A.

10 Configuration specification

Chapter 10.1 specifies the structure (containers) and the parameters of the module CRYIF.

Chapter 10.2 specifies additionally published information of the module CRYIF.

10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.
Note: The Ids in the configuration containers shall be consecutive, gapless and shall start from zero.

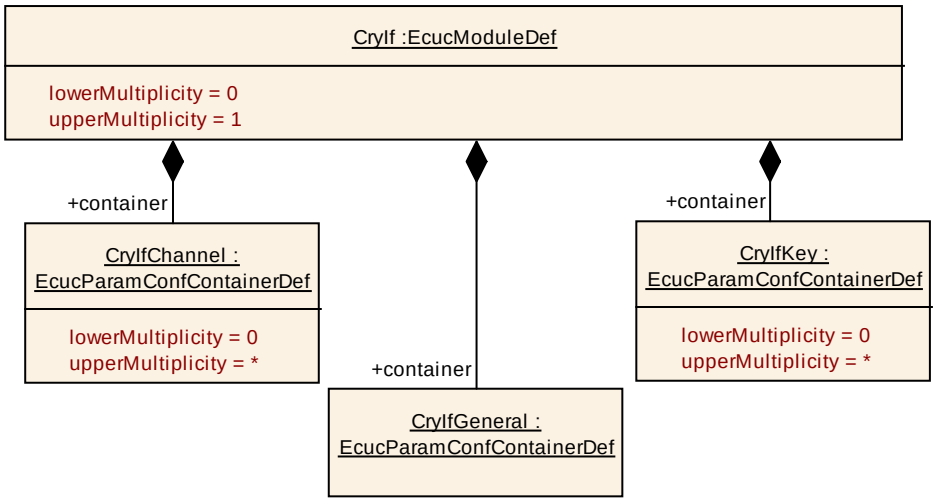
10.1.1 Variants

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

10.1.2 CryIf

SWS Item	**ECUC_CryIf_00001 :**
Module Name	*CryIf*
Module Description	Configuration of the Crypto Interface.
Post-Build Variant Support	false

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CryIfChannel	0..*	Container for incorporation of CryIfChannel.
CryIfGeneral	1	Container for incorporation of CryIfGeneral.
CryIfKey	0..*	Container for incorporation of CryIfKey.



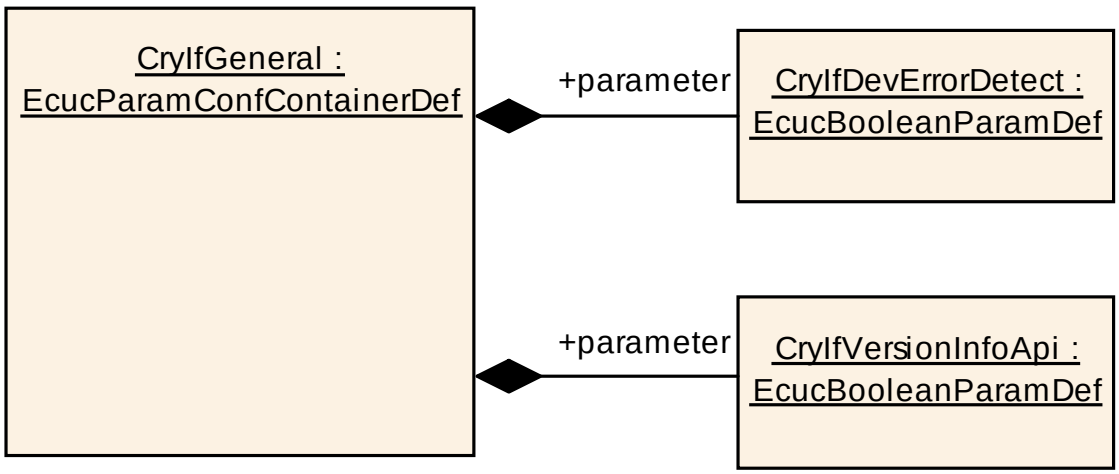
10.1.3 CryIfGeneral

SWS Item	ECUC_CryIf_00009 :
Container Name	CryIfGeneral
Description	Container for incorporation of CryIfGeneral.
Configuration Parameters	

SWS Item	ECUC_CryIf_00010 :
Name	CryIfDevErrorDetect
Description	Switches the development error detection and notification on or off. true: detection and notification is enabled. false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--
Scope / Dependency	scope: local

SWS Item	ECUC_CryIf_00011 :
Name	CryIfVersionInfoApi
Description	Pre-processor switch to enable and disable availability of the API CryIf_GetVersionInfo(). True: API CryIf_GetVersionInfo() is available False: API CryIf_GetVersionInfo() is not available.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	--
Scope / Dependency	scope: local

No Included Containers



10.1.4 CryIfChannel

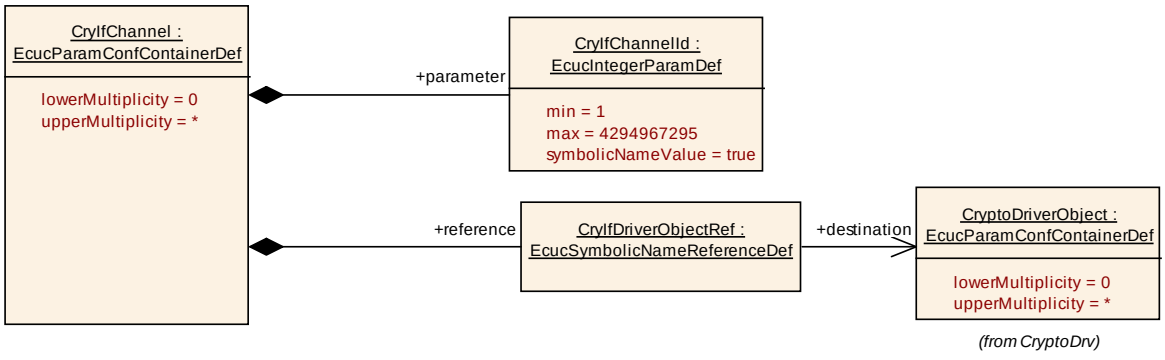
SWS Item	ECUC_CryIf_00002 :
Container Name	CryIfChannel

Description	Container for incorporation of CryIfChannel.
Configuration Parameters	

SWS Item	**ECUC_CryIf_00004 :**
Name	CryIfChannelId
Description	Identifier of the crypto channel. Specifies to which crypto channel the CSM queue is connected to.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	1 .. 4294967295
Default value	--
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false
Scope / Dependency	scope: local

SWS Item	**ECUC_CryIf_00005 :**
Name	CryIfDriverObjectRef
Description	This parameter refers to a Crypto Driver Object. Specifies to which Crypto Driver Object the crypto channel is connected to
Multiplicity	1
Type	Symbolic name reference to [CryptoDriverObject]
Post-Build Variant Multiplicity	false
Post-Build Variant Value	false
Scope / Dependency	scope: local

No Included Containers



10.1.5 CryIfKey

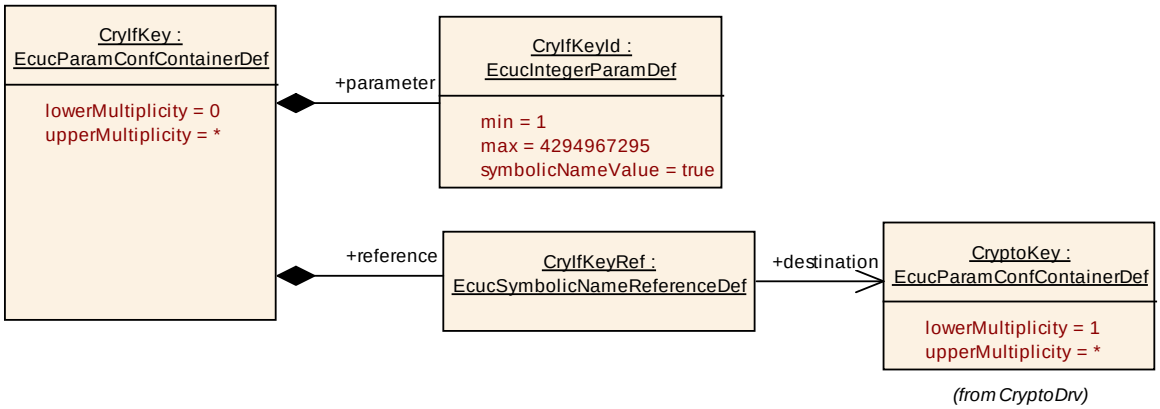
SWS Item	**ECUC_CryIf_00003 :**
Container Name	CryIfKey
Description	Container for incorporation of CryIfKey.
Configuration Parameters	

SWS Item	**ECUC_CryIf_00007 :**
Name	CryIfKeyId
Description	Identifier of the CryIf key. Specifies to which CryIf key the CSM key is mapped to.
Multiplicity	1

Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	1 .. 4294967295
Default value	--
Post-Build Variant Value	false
Scope / Dependency	scope: local

SWS Item	**ECUC_CryIf_00008 :**
Name	CryIfKeyRef
Description	This parameter refers to the crypto driver key. Specifies to which crypto driver key the CryIf key is mapped to.
Multiplicity	1
Type	Symbolic name reference to [CryptoKey]
Post-Build Variant Value	false
Scope / Dependency	scope: local

No Included Containers



(from CryptoDrv)

10.2 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

Additional module-specific published parameters are listed below if applicable.