VECTOR

MICROSAR CSM

Technical Reference

Cryptographic Service Manager

Version 1.03.00

Authors	Markus Schneider, Anant Gupta
Status	Released

Document Information

History

Author	Date	Version	Remarks
Schneider, Markus	2017-03-14	1.01.00	Initial Creation for ASR 4.3
Gupta, Anant	2017-03-17	1.01.00	Add configurational details
Schneider, Markus	2017-05-08	1.02.00	3.1.1 Added backward compatible static definition limitation 5 Adapted to Specification
Schneider, Markus	2017-06-07	1.03.00	Adapted chapter 4.2 Added information about SecureCounter Removed API description for SecureCounter

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_CryptoServiceManager.pdf	4.3.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	4.3.0

Contents

Illustrations

Tables

1 Component History

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

Component Version	New Features
1.00	Initial beta release
1.01	Adaptions to the specification; several improvements and bugfixes
1.02	Serial Production Release
1.03	SafeBsw Release

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4.3	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CSM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CSM_MODULE_ID	110 decimal

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

The CSM provides synchronous and asynchronous services to enable a unique access to basic cryptographic functionalities for software components (SWC) and basic software (BSW). The CSM offers a standardized interface to higher software layers to access these functionalities.

2.1 Architecture Overview

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

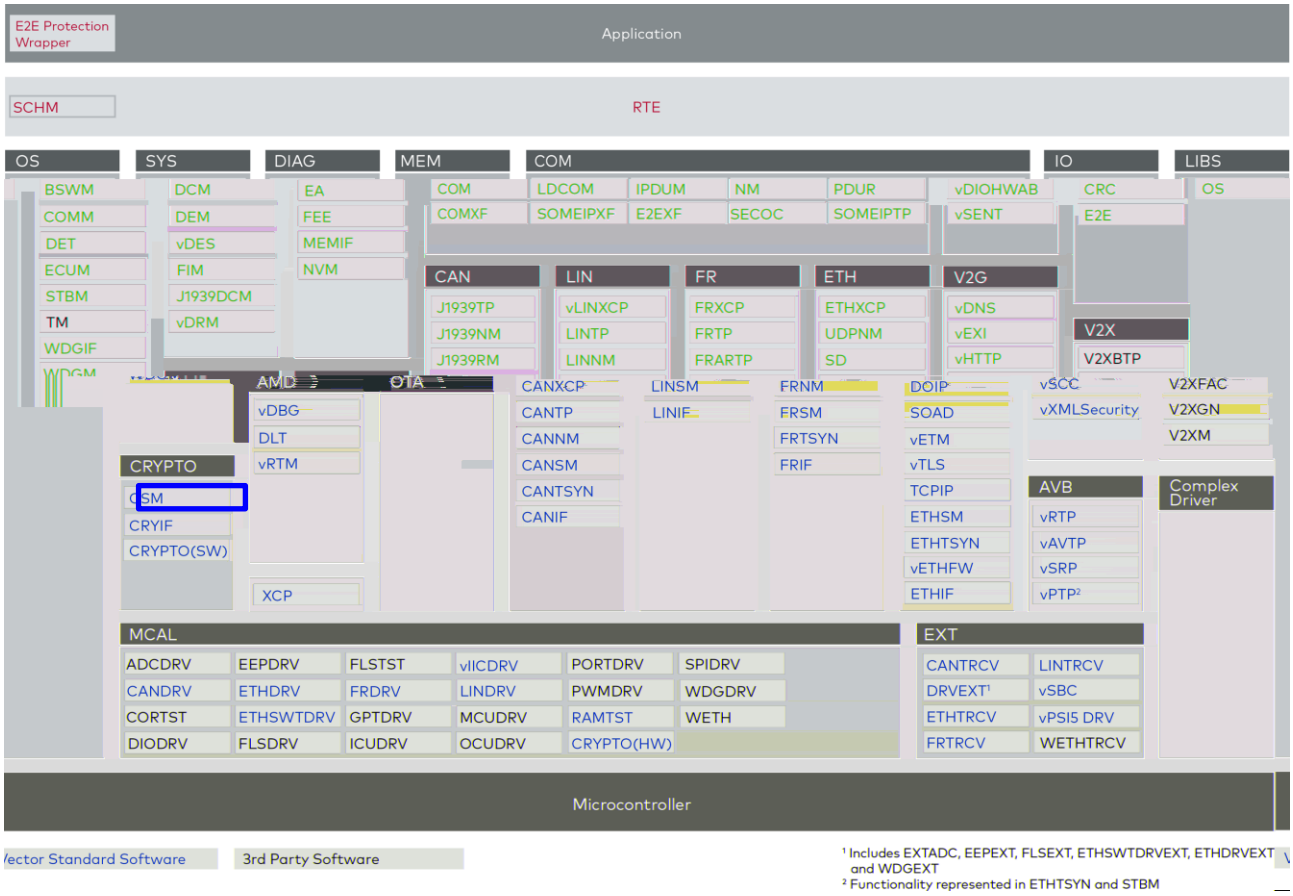

Figure 2-1 AUTOSAR 4.3 Architecture Overview

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

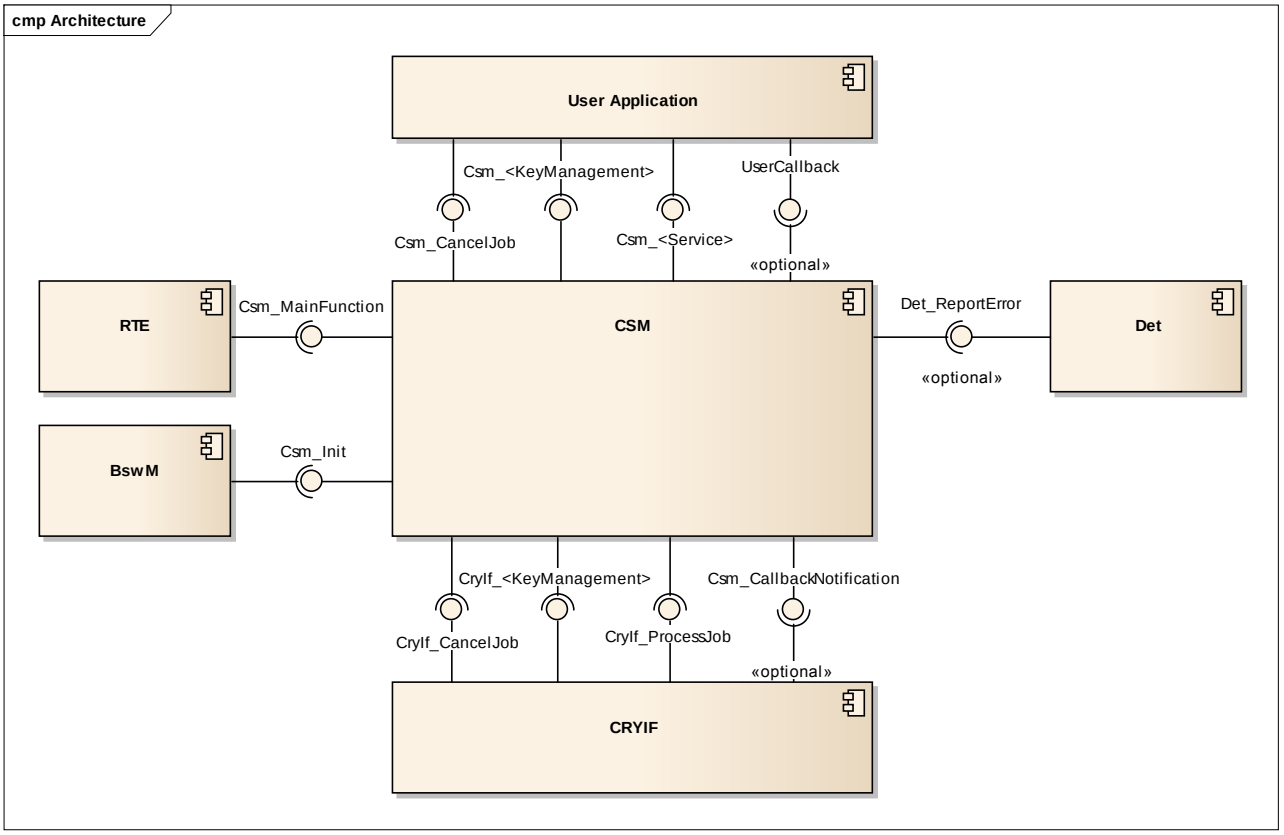

Figure 2-2 Interfaces to adjacent modules of the CSM

3 Functional Description

3.1 Features

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

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

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Queueing and cancellation of jobs
Job prioritization
Synchronous and asynchronous job handling
Key management APIs

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

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

Not Supported AUTOSAR Standard Conform Features
4.x backward compatible API and Client-Server Interface
4.x backward compatible definitions are static and will be removed in the next release
Secure Counter Increment and Secure Counter Read are not supported

Table 3-2 Not supported AUTOSAR standard conform features

3.2 Initialization

Before any other functionality of the CSM module can be used the initialization function `Csm_Init()` has to be called by the BSWM.

For manual null initialization of RAM variables the CSM offers the function `Csm_InitMemory()` which can be called before the `Csm_Init()`.

3.3 Main Functions

The CSM module implementation provides one main function. When the usage of asynchronous job processing is enabled, this main function has to be called cyclically on task level. The main function is responsible to dispatch new jobs to the underlying CRYIF respectively CRYPTO driver.

Service ID	Service
0x6F	Csm_CancelJob
0x70	Csm_CallbackNotification

Table 3-3 Service IDs

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

Error Code	Description
0x01	API request called with invalid parameter (Nullpointer)
0x05	API request called before initialization of CSM module
0x07	Initialization of CSM module failed
0x09	Requested service is not initialized
0x03	API request called with invalid parameter (invalid method for selected service)
0x11	The service Csm_Init() is called while the module is already initialized

Table 3-4 Errors reported to DET

4 Integration

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

4.1 Scope of Delivery

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

4.1.1 Static Files

File Name	Description
Csm.c	This is the source file of the CSM.
Csm.h	This is the header file of the CSM.
Csm_Cbk.h	This is the header file which contains the callback function declaration.
Csm_Types.h	This is the common header file used within the crypto stack.

Table 4-1 Static files

4.1.2 Dynamic Files

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

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

Table 4-2 Generated files

4.2 Critical Sections

The CSM has critical code sections which must not be interrupted. The sections are related to the queue handling, job sorting and job queuing.

The CSM module calls the following function when entering a critical section:

> SchM_Enter_Csm_CSM_EXCLUSIVE_AREA_0

When the critical section is left the following function is called by the CSM module:

> SchM_Exit_Csm_CSM_EXCLUSIVE_AREA_0

This critical section is needed to ensure consistency of global RAM variables regarding the queue handling. The runtime and length of the critical section depends on the configured queue size.

Note
The critical section is only needed when asynchronous jobs are configured.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

Application related types defined by the CSM are described in this chapter. All types not described here are defined by the CSM as described in [1].

Type Name	C-Type	Description	Value Range
Crypto_OperationModeType	uint8	Indicator of the mode(s)/operation(s) to be performed.	CRYPTO_OPERATIONMODE_START Perform start operation
			CRYPTO_OPERATIONMODE_UPDATE Perform update operation
			CRYPTO_OPERATIONMODE_FINISH Perform finish operation
			CRYPTO_OPERATIONMODE_STREAMSTART Perform start and update operation
			CRYPTO_OPERATIONMODE_SINGLECALL Perform start, update and finish operation
Crypto_VerifyResultType	uint8	Enumeration of the result type of verification operations.	CRYPTO_E_VER_OK The result of the verification is "true", i.e. the two compared elements are identical. This return code shall be given as value "0"
			CRYPTO_E_VER_NOT_OK 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".

Table 5-1 Type definitions

5.2 Services provided by CSM

5.2.1 Csm_Init

Prototype	
void (void)	
Parameter	
void	none

Return code	
void	none
Functional Description	
Initializes the Csm.	
Particularities and Limitations	
None Set all service states to initial idle.	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-2 Csm_Init

5.2.2 Csm_InitMemory

Prototype	
`void (void)`	
Parameter	
void	none
Return code	
void	none
Functional Description	
Power-up memory initialization.	
Particularities and Limitations	
Use this function in case these variables are not initialized by the startup code. Module is uninitialized. Initialize component variables at power up.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-3 Csm_InitMemory

5.2.3 Csm_GetVersionInfo

Prototype	
`void (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	
Implements the API to be called cyclically to process the requested services.	
Particularities and Limitations	
> Csm is initialized.GetVersionInfo API is enabled via pre-compile configuration. Calls the configured main function of the specific Cry service if requested.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-4 Csm_GetVersionInfo

5.2.4 Csm_CancelJob

Prototype	
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Not used, just for interface compatibility provided
Return code	
Std_ReturnType	CSM_E_OK Request successful
Std_ReturnType	CSM_E_NOT_OK Request failed
Functional Description	
Cancels the given job.	
Particularities and Limitations	
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.	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-5 Csm_CancelJob

5.2.5 Csm_KeyElementSet

Prototype	
`Std_ReturnType (uint32 keyId, uint32 keyElementId, const uint8 *keyPtr, uint32 keyLength)`	
Parameter	
keyId [in]	Holds the identifier of the job using the CSM service.
keyElementId [in]	Holds the identifier of the key element to be written.
keyPtr [in]	Holds the pointer to the key element bytes to be processed.
keyLength [in]	Contains the number of key element bytes.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service 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 because key element size does not match size of provided data
Functional Description	
Sets the given key element bytes to the key identified by keyId.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Non-Reentrant	

Table 5-6 Csm_KeyElementSet

5.2.6 Csm_KeySetValid

Prototype	
`Std_ReturnType (uint32 keyId)`	
Parameter	
keyId [in]	Holds the identifier of the key for which a new material shall be validated.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is busy

Functional Description
Sets the key state of the key identified by keyId to valid.
Particularities and Limitations
-
Call context
> TASK > This function is Synchronous > This function is Reentrant

Table 5-7 Csm_KeySetValid

5.2.7 Csm_KeyElementGet

Prototype	
`Std_ReturnType (uint32 keyId, uint32 keyElementId, uint8 *keyPtr, uint32 *keyLengthPtr)`	
Parameter	
keyId [in]	Holds the identifier of the key from which a key element shall be extracted.
keyElementId [in]	Holds the identifier of the key element to be extracted.
keyPtr [out]	Holds the pointer to the memory location where the key shall be copied to.
inout] [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.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is busy
	CRYPTO_E_KEY_READ_FAIL Request failed because read access was denied
	CRYPTO_E_KEY_NOT_AVAILABLE Request failed because the key is not available
	CRYPTO_E_SMALL_BUFFER The provided buffer is too small to store the result
Functional Description	
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.	
Particularities and Limitations	
-	
Call context	
> TASK	

> This function is Synchronous > This function is Reentrant

Table 5-8 Csm_KeyElementGet

5.2.8 Csm_KeyElementCopy

Prototype	
`Std_ReturnType (uint32 keyId, uint32 keyElementId, uint32 targetKeyId, uint32 targetKeyElementId)`	
Parameter	
keyId [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.
targetKeyId [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	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy
	CRYPTO_E_KEY_READ_FAIL Request failed because read access was denied
	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 because key element sizes are not compatible
Functional Description	
This function shall copy a key elements from one key to a target key.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-9 Csm_KeyElementCopy

5.2.9 Csm_KeyCopy

Prototype	
`Std_ReturnType (uint32 keyId, uint32 targetKeyId)`	
Parameter	
keyId [in]	Holds the identifier of the key whose key element shall be the source element.
targetKeyId [in]	Holds the identifier of the key whose key element shall be the destination element.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy
	CRYPTO_E_KEY_READ_FAIL Request failed because read access was denied
	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 because key element sizes are not compatible
Functional Description	
nction shall copy all key elements from the source key to a target key.	
ularities and Limitations	
ontext	
SK	
s function is Synchronous	

Return code	
Std_ReturnType	CSM_E_OK Request successful
Std_ReturnType	CSM_E_NOT_OK Request failed
Functional Description	
Feeds a key with a random seed.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-11 Csm_RandomSeed

5.2.11 Csm_KeyGenerate

Prototype	
`Std_ReturnType (uint32 keyId)`	
Parameter	
keyId [in]	Holds the identifier of the key for which a new material shall be generated.
Return code	
Std_ReturnType	CSM_E_OK Request successful
Std_ReturnType	CSM_E_NOT_OK Request failed
Functional Description	
Generates a key based on key element input.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-12 Csm_KeyGenerate

5.2.12 Csm_KeyDerive

Prototype
`Std_ReturnType (uint32 keyId, uint32 targetKeyId)`

Parameter	
keyId [in]	Holds the identifier of the key used for key derivation.
targetKeyId [in]	Holds the identifier of the key which is used to store the derived key.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, Crypto Driver Object is busy
Functional Description	
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.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-13 Csm_KeyDerive

5.2.13 Csm_KeyExchangeCalcPubVal

Prototype	
`Std_ReturnType (uint32 keyId, uint8 *publicValuePtr, uint32 *publicValueLengthPtr)`	
Parameter	
keyId [in]	Holds the key identifier of the key to be used for the key exchange protocol.
publicValuePtr [out]	Contains the pointer to the data where the public value shall be stored.
inout] [out]	publicValueLengthPtr Holds a pointer to the memory location in which the public value length in bytes is stored. On calling this function, this parameter shall contain the size of the buffer in bytes provided by publicValuePtr. When the request has finished, the actual length of the returned value shall be stored.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_KEY_NOT_VALID Request failed because the key's state is "invalid"
	CRYPTO_E_SMALL_BUFFER The provided buffer is too small to store the result
Functional Description	
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.
Particularities and Limitations
-
Call context
> TASK > This function is Synchronous > This function is Reentrant

Table 5-14 Csm_KeyExchangeCalcPubVal

5.2.14 Csm_KeyExchangeCalcSecret

Prototype	
`Std_ReturnType (uint32 keyId, const uint8 *partnerPublicValuePtr, uint32 partnerPublicValueLength)`	
Parameter	
keyId [in]	Holds the key identifier of the key to be used for the key exchange protocol.
partnerPublicValuePtr [in]	Holds the pointer to the memory location containing the partner's public value.
partnerPublicValueLength [in]	Contains the number of bytes of the partner public value.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_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
Functional Description	
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.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-15 Csm_KeyExchangeCalcSecret

5.2.15 Csm_CertificateParse

Prototype	
`Std_ReturnType (uint32 keyId)`	
Parameter	
keyId [in]	Holds the identifier of the key to be used for the certificate parsing.
Return code	
Std_ReturnType	CSM_E_OK Request successful
Std_ReturnType	CSM_E_NOT_OK Request failed
Functional Description	
This function shall dispatch the certificate parse function to the CRYIF.	
Particularities and Limitations	
-	
Call context	
> TASK > This function is Synchronous > This function is Reentrant	

Table 5-16 Csm_CertificateParse

5.2.16 Csm_CertificateVerify

Prototype	
`Std_ReturnType (uint32 keyId, uint32 verifyKeyId, Crypto_VerifyResultType *verifyPtr)`	
Parameter	
keyId [in]	Holds the identifier of the key to be used to validate the certificate.
verifyKeyId [in]	Holds the identifier of the key containing the certificate to be verified.
verifyPtr [in]	Holds a pointer to the memory location, which will contain the result of the certificate verification.
Return code	
Std_ReturnType	CSM_E_OK Request successful
Std_ReturnType	CSM_E_NOT_OK Request failed
Functional Description	
Verifies the certificate stored in the key referenced by verifyKeyId with the certificate stored in the key referenced by keyId.	
Particularities and Limitations	
-	
Call context	
> TASK	

> This function is Synchronous > This function is Reentrant

Table 5-17 Csm_CertificateVerify

5.2.17 Csm_Hash

Prototype	
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, uint8 *resultPtr, uint32 *resultLengthPtr)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data for which the hash shall be computed.
dataLength [in]	Contains the number of bytes to be hashed.
resultPtr [out]	Contains the pointer to the data where the hash value shall be stored.
resultLengthPtr [in,out]	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.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
This interface shall be used for the hash computation service.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-18 Csm_Hash

5.2.18 Csm_MacGenerate

Prototype	
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, uint8 *macPtr, uint32 *macLengthPtr)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data for which the MAC shall be computed.
dataLength [in]	Contains the number of bytes for the MAC generation.
macPtr [out]	Contains the pointer to the data where the MAC shall be stored.
macLengthPtr [in,out]	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.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
Uses the given data to perform a MAC generation and stores the MAC in the memory location pointed to by the MAC pointer.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-19 Csm_MacGenerate

5.2.19 Csm_MacVerify

Prototype	
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, const uint8 *macPtr, uint32 macLength, Crypto_VerifyResultType *verifyPtr)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.

mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data for which the MAC shall be verified.
dataLength [in]	Contains the number of data bytes for which the MAC shall be verified.
macPtr [in]	Holds a pointer to the MAC to be verified.
macLength [in]	Contains the MAC length in BITS to be verified.
verifyPtr [out]	Holds a pointer to the memory location, which will hold the result of the MAC verification.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
Verifies the given MAC by comparing if the MAC is generated with the given data.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-20 Csm_MacVerify

5.2.20 Csm_Encrypt

Prototype
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, uint8 *resultPtr, uint32 *resultLengthPtr)`
Parameter

jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data to be encrypted.
dataLength [in]	Contains the number of bytes to encrypt.
resultPtr [out]	Contains the pointer to the data where the encrypted data shall be stored.
resultLengthPtr [in,out]	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.

Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
Encrypts the given data and stores the ciphertext in the memory location pointed by the result pointer.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-21 Csm_Encrypt

5.2.21 Csm_Decrypt

Prototype
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, uint8 *resultPtr, uint32 *resultLengthPtr)`

Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data to be decrypted.
dataLength [in]	Contains the number of bytes to decrypt.
resultPtr [out]	Contains the pointer to the data where the decrypted data shall be stored.
resultLengthPtr [in,out]	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.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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

Functional Description
Decrypts the given encrypted data and stores the decrypted plaintext in the memory location pointed by the result pointer.
Particularities and Limitations
Service is idle. -
Call context
> TASK > This function is Reentrant

Table 5-22 Csm_Decrypt

5.2.22 Csm_AEADEncrypt

Prototype

```
Std_ReturnType                  (uint32 jobId, Crypto_OperationModeType mode,
const uint8 *plaintextPtr, uint32 plaintextLength, const uint8
*associatedDataPtr, uint32 associatedDataLength, uint8 *ciphertextPtr, uint32
*ciphertextLengthPtr, uint8 *tagPtr, uint32 *tagLengthPtr)
```

Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
plaintextPtr [in]	Contains the pointer to the data to be encrypted.
plaintextLength [in]	Contains the number of bytes to encrypt.
associatedDataPtr [in]	Contains the pointer to the associated data.
associatedDataLength [in]	Contains the number of bytes of the associated data.
ciphertextPtr [out]	Contains the pointer to the data where the encrypted data shall be stored.
ciphertextLengthPtr [in,out]	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.
tagPtr [out]	Contains the pointer to the data where the Tag shall be stored.
tagLengthPtr [in,out]	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.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
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.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-23 Csm_AEADEncrypt

5.2.23 Csm_AEADDecrypt

Prototype	

`Std_ReturnType`
 	
`_VerifyResultType *verifyPtr)`	

Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
ciphertextPtr [in]	Contains the pointer to the data to be decrypted.
ciphertextLength [in]	Contains the number of bytes to decrypt.
associatedDataPtr [in]	Contains the pointer to the associated data.
associatedDataLength [in]	Contains the number of bytes of the associated data.
tagPtr [in]	Contains the pointer to the data where the Tag shall be stored.
tagLength [in]	Contains the length in bytes of the Tag to be verified.
plaintextPtr [out]	Contains the pointer to the data where the encrypted data shall be stored.
plaintextLengthPtr [in,out]	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.
verifyPtr [out]	Contains the pointer to the result of the verification.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
Uses the given data to perform an AEAD decryption and stores the plaintext and the result of in the memory locations pointed by the plaintext pointer and verifyPtr pointer.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-24 Csm_AEADDecrypt

5.2.24 Csm_SignatureGenerate

Prototype	
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, uint8 *resultPtr, uint32 *resultLengthPtr)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data to be signed.
dataLength [in]	Contains the number of bytes to sign.
resultPtr [out]	Contains the pointer to the data where the signature shall be stored.
resultLengthPtr [in,out]	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 in bytes provided by resultPtr. When the request has finished, the actual length of the returned value shall be stored.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.	
Particularities and Limitations	
Service is idle. -	

Call context
> TASK
> This function is Reentrant

Table 5-25 Csm_SignatureGenerate

5.2.25 Csm_SignatureVerify

Prototype	
`Std_ReturnType (uint32 jobId, Crypto_OperationModeType mode, const uint8 *dataPtr, uint32 dataLength, const uint8 *signaturePtr, uint32 signatureLength, Crypto_VerifyResultType *verifyPtr)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
mode [in]	Indicates which operation mode(s) to perfom.
dataPtr [in]	Contains the pointer to the data to be verified.
dataLength [in]	Contains the number of bytes to be verified.
signaturePtr [in]	Holds a pointer to the signature to be verified.
signatureLength [in]	Contains the signature length in bytes.
verifyPtr [out]	Holds a pointer to the memory location, which will hold the result of the signature verification.
Return code	
Std_ReturnType	CSM_E_OK Request successful
	CSM_E_NOT_OK Request failed
	CRYPTO_E_BUSY Request failed, service is 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
Functional Description	
Verifies the given signature by comparing with a generated signature.	
Particularities and Limitations	
Service is idle. -	
Call context	
> TASK > This function is Reentrant	

Table 5-26 Csm_SignatureVerify

5.2.26 Csm_RandomGenerate

Prototype	
`Std_ReturnType (uint32 jobId, uint8 *resultPtr, uint32 *resultLengthPtr)`	
Parameter	
jobId [in]	Holds the identifier of the job using the CSM service.
resultPtr [out]	Holds a pointer to the memory location which will hold the result of the random number generation.
resultLengthPtr [in,out]	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.

Component	API
DET	Det_ReportError

Table 5-28 Services used by the CSM

5.4 Callback Functions

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

5.4.1 Csm_CallbackNotification

Prototype	
`void ( 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 CSM 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 5-29 Csm_CallbackNotification

5.5 Service Ports

5.5.1 Client Server Interface

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

5.5.1.1 Provide Ports on CSM Side

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

6 Configuration

6.1 Overview

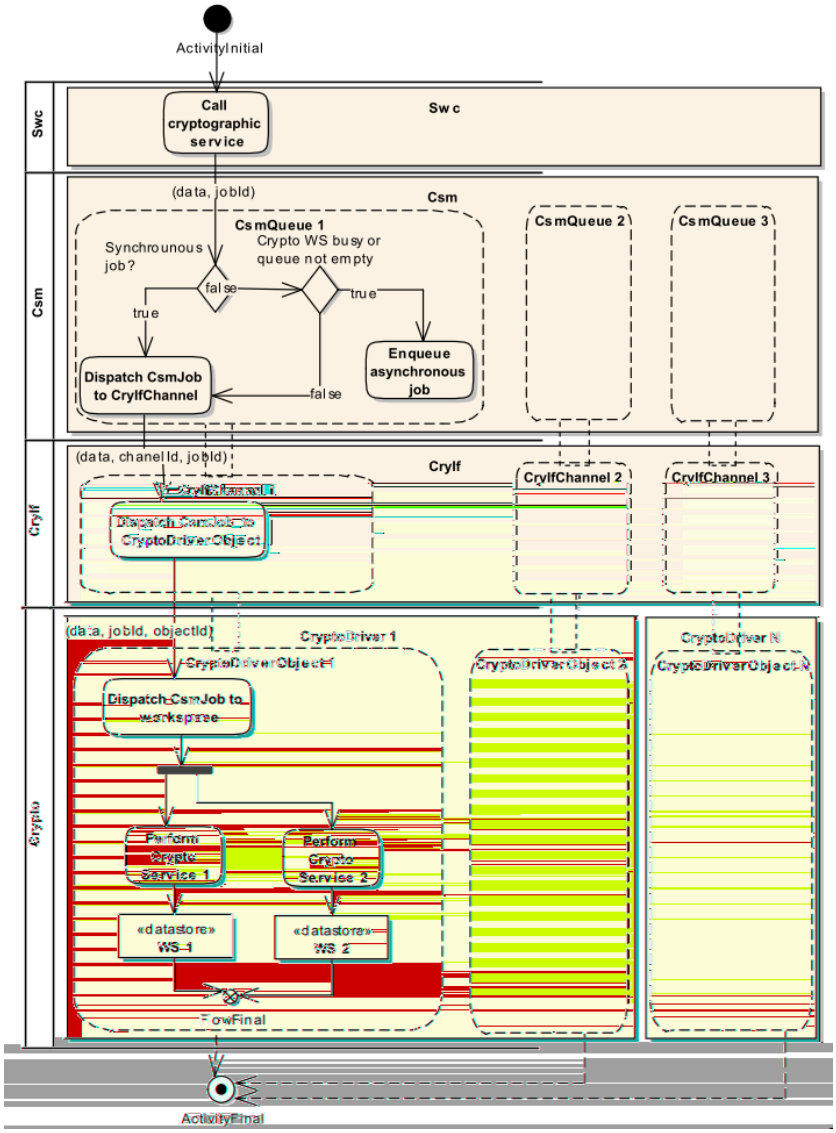

Figure 6-1 Structural overview and basic workflow

Figure 6-1 gives a structural overview and shows the basic workflow of the crypto stack. The Swc requests a cryptographic service of the CSM providing the necessary data and the id of the CsmJob which will be performed in the underlying CryptoDriverObject. The CsmJob contains all computational parameters for performing the cryptographic operation (see chapter 6.2 and 6.5.6).

At configuration time, each CsmJob is assigned to a CsmQueue, regardless of processing type (synchronous or asynchronous) of the job. Each CsmQueue is mapped to CryIfQueue and a CryIfQueue itself is mapped to a CryptoDriverObject of a crypto driver. At the crypto layer, N crypto drivers may coexist and do not interfere each other.

The CryptoDriverObject must allocate the required workspace and support the cryptographic operational in general. A job in computation occupies the cryptographic workspace of the cryptographic service; therefore jobs running in the same crypto driver object are processed in serialized fashion, whereas jobs running in different crypto driver objects or jobs of different cryptographic type may run in parallel.

All CSM attributes can be configured with the following tools:

> Configuration in DaVinci Configurator 5 Pro for a detailed description see 6.5

6.2 Configuration of a cryptographic operation

A cryptographic algorithm is mostly configurable and must be parametrized to operate in specific mode. A CsmJob encapsulates all these information and can be considered as an instance of a cryptographic algorithm realized in a CryptoDriverObject. The cryptographic operation and the operational mode are configured in a CsmPrimitive and is referred by a concrete CsmJob. A CsmPrimitive can be of following primitive service types:

- MacGenerate
- MacVerifiy
- SignatureGenerate
- SignatureVerify
- Encrypt
- Decrypt
- Hash
- Random Generate
- AEADEncrypt
- AEADDecrypt
- Secure Counter Read
- Secure Counter Increment

Further, each CsmPrimitive defines the algorithm family, mode and secondary family. The user has to ensure that the underlying destination crypto driver object supports the configured combination of these three parameters. The CryptoDriverObject itself must refer at least one CryptoPrimitive which matches in primitive service type, family, mode and secondary family (compare Figure 6-2, Figure 6-3 and Figure 6-4).

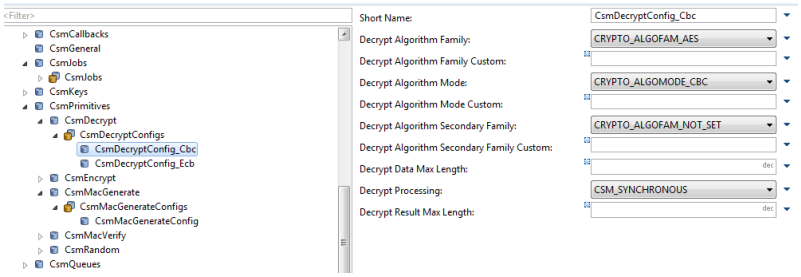

Figure 6-2 Configuration CsmPrimitive

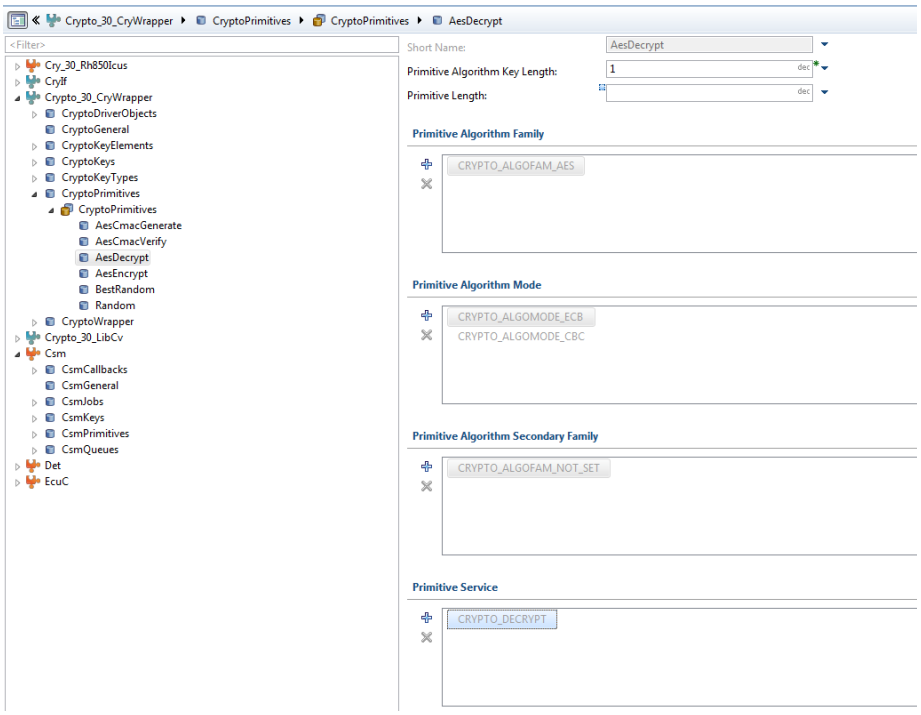

Figure 6-3 Configuration CryptoPrimitive

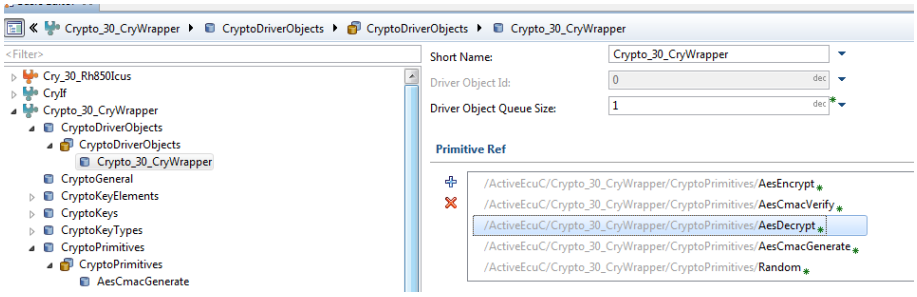

Figure 6-4 Configuration CryptoDriverObject

6.3 Configuration Variants

The CSM supports the configuration variants

> VARIANT-PRE-COMPILE

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

6.4 Backward Compatibility

The CSM does not support backward compatible interfaces to AUTOSAR 4.2.x, but offers old standard definitions within the `Csm_Types.h`. However, these definitions are not configurable and must be adapted manually if needed.

6.5 Configuration with DaVinci Configurator 5 Pro

The CSM is configured with the help of the configuration tool DaVinci Configurator 5 Pro.

6.5.1 General Properties

Attribute Name	Values Default value is typed bold	Description
CsmDevErrorDetect	**TRUE** FALSE	Pre-processor switch to enable and disable development error detection.
CsmVersionInfoApi	**TRUE** FALSE	Pre-processor switch to enable and disable availability of the API Csm_GetVersionInfo(). - True: API Csm_GetVersionInfo() is available. - False: API Csm_GetVersionInfo() is not available.
CsmCustomIncludeFiles	-	Include custom header file
CsmMainFunctionPeriod	0 - 3600 **0.01**	Specifies the period of main function Csm_MainFunction in seconds.
CsmAsymPublicKeyMaxLength	1 - 4294967295	Maximum length in bytes of an asymmetric public key for all algorithms
CsmAsymPrivateKeyMaxLength	1 - 4294967295	Maximum length in bytes of an asymmetric private key for all algorithms
CsmSymKeyMaxLength	1 - 4294967295	Maximum length in bytes of a symmetric key for all algorithms
CsmUseDeprecated	TRUE **FALSE**	Currently not supported Decides if the deprecated interfaces shall be used (Backwards compatibility). - True: use deprecated interfaces. - False: use normal interfaces.

Table 6-1 General Properties

6.5.2 Key Properties

Attribute Name	Values Default value is typed bold	Description
CsmKeyId	0 - 4294967295	Identifier of the CsmKey
CsmKeyUsePort	**TRUE** FALSE	Does the key need RTE interfaces? - True: RTE interfaces used for this key - False: No RTE interfaces used for this key
CsmKeyRef	-	This parameter refers to the used CryIfKey. The underlying CryIfKey refers to a specific CryptoKey in the Crypto Driver.

Table 6-2 Key Properties

6.5.3 Primitives Properties

Attribute Name	Values Default value is typed bold	Description
Csm<Service>Algorithm Mode	Service depending literals	Determines the algorithm mode used for the crypto service
Csm<Service>Processing	CSM_ASYNCHRONOUS CSM_SYNCHRONOUS	Determines how the interface shall be used for that primitive

Attribute Name	Values Default value is typed bold	Description
		operation has finished
CsmCallbackId	0 - 4294967295	Identifier of the callback function.

Table 6-5 Callback Properties

6.5.6 Job Properties

Attribute Name	Values Default value is typed bold	Description
CsmJobPriority	0 - 4294967295	Priority of the job. The higher the value, the higher the job's priority.
CsmJobId	0 - 4294967295	Identifier of the CSM job
CsmJobUsePort	**TRUE** FALSE	Does the job need RTE interfaces? - True: the job needs RTE interfaces - False: the job needs no RTE interfaces
CsmJobPrimitiveCallbackUpdateNotification	TRUE **FALSE**	This parameter indicates, whether the callback function shall be called, if the UPDATE operation has finished.
CsmJobUseOldPort	TRUE **FALSE**	Does the user needs old RTE interfaces?
CsmJobPrimitiveRef	-	This parameter refers to the used CsmPrimitive. Different jobs may refer to one CsmPrimitive. The referred CsmPrimitive provides detailed information on the actual cryptographic routine.
CsmJobPrimitiveCallbackRef	-	This parameter refers to the used CsmCallback. The referred CsmCallback is called when the crypto job has been finished.
CsmJobQueueRef	-	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.
CsmJobKeyRef	-	This parameter refers to the key which shall be used for the CsmPrimitive. It's possible to use a CsmKey for different jobs

Table 6-6 Job Properties

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
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

Table 7-2 Abbreviations

8 Contact

Visit our website for more information on

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

www.vector.com