Document Title	Specification of Chinese Vehicle-2-X Security
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	1032

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• Initial release

Contents

1 Introduction and functional overview

This document specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Chinese Vehicle-2-X Security (CnV2xSec).

The Chinese Vehicle-2-X Security (CnV2xSec) together with the Chinese Vehicle-2-X Message (CnV2xMsg), Chinese Vehicle-2-X Management (CnV2xM), Chinese Vehicle-2-X Network (CnV2xNet) forms the Chinese V2X stack within the AUTOSAR architecture.

The bases for this document are the Chinese LTE-V2X based standards [1][2]. It is assumed that the reader is familiar with these standards.

1.1 Architecture Overview

Positioning of the CnV2xSec module within the AUTOSAR BSW and the Layered Software architecture is shown as below.

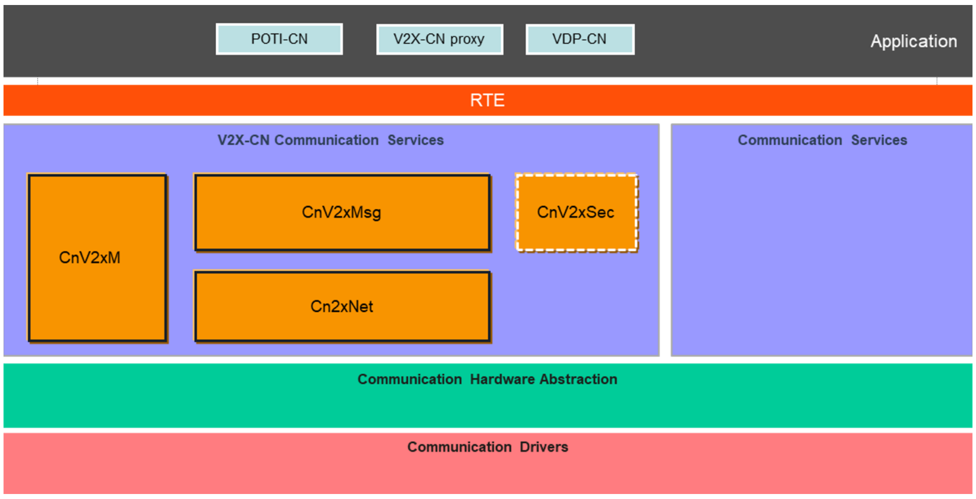

Figure 1.1: AUTOSAR BSW software architecture - CnV2xSec scope

The CnV2xSec module provides services of message encapsulation, decapsulation and pseudonym management.

1.2 Functional Overview

The CnV2xSec module provides standardized security services to the V2X-CN Stack according to CCSA Specifications, the CnV2xSec module should comply with the standards [1], [2] and [3]. The APIs shall be implemented using Csm and NvM services provided by AUTOSAR.

2 Acronyms and Abbreviations

Abbreviation / Acronym:	Description:
AID	Application identifier
BS	Basic Service
BSM	Basic safety Message
C-V2X	Cellular based Vehicle to Everything
CCSA	China Communications Standards Association
CnV2xMsg	Chinese Vehicle-2-X Message
CnV2xM	Chinese Vehicle-2-X Management
CnV2xNet	Chinese Vehicle-2-X Network
CnV2xSec	Chinese Vehicle-2-X Security
DE	Data Element
DF	Data Frame
EcuM	Electronic Control Unit Manager
NTCAS	National Technical Committee of Auto Standardization
PH	Path History
POTI	Position and Time
RSI	Road Side Information
RSM	Road Side Message
RSU	Road Side Unit
SPAT	Signal Phase And Time
SPDU	Secured protocol data Unit
VDP	Vehicle Data Provider

3 Related documentation

3.1 Input documents & related standards and norms

[1] GB/T: Technical requirements and test methods of vehicular communication system based on LTE-V2X direct communication (Draft Edition: 2022-04-01)
http://www.catarc.org.cn/

[2] YD/T 3957-2021: Technical Requirement of Security Certificate Management System for LTE-based Vehicular Communication
http://www.ccsa.org.cn/

[3] YD/T 3594-2019: General technical requirements of Security for Vehicular Communication based on LTE
http://www.ccsa.org.cn/

[4] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[5] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[6] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[7] Specification of NVRAM Manager
AUTOSAR_CP_SWS_NVRAMManager

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules (SWS BSW General) [4], which is also valid for CnV2xSec.

Thus, the specification SWS BSW General [4] shall be considered as additional and required specification for CnV2xSec.

4 Constraints and assumptions

4.1 Limitations

The V2X modules follow the technical requirements regarding the Day-1 scenarios defined by CCSA and NTCAS.

4.2 Known Limitations in AUTOSAR R22-11

The unicast service is not supported in V2X Day-1 scenarios. So the encryption and decryption of unicast messages are not implemented in AUTOSAR R22-11.

4.3 Applicability to car domains

This specification is applicable to all car domains.

5 Dependencies to other modules

This section describes the relations of the CnV2xSec module to other modules within the AUTOSAR basic software architecture. It outlines the modules that are required or optional for the realization of the CnV2xSec module and the CnV2xSec services that these modules use.

5.1 AUTOSAR Default Error Tracer (DET)

In development mode, CnV2xSec module reports errors through the Det_ReportError function of the DET Module [5].

5.2 AUTOSAR Ecu State Manager (EcuM)

The EcuM [6] initializes the CnV2xSec module by calling CnV2xSec_Init specified in 8.3.1.

5.3 AUTOSAR CnV2xMsg

The CnV2xMsg module is used by CnV2xSec to get time, distance and vehicleEvent-Flags.

5.4 AUTOSAR Csm

The Csm module is used for cryptographic calculations, needed by the CnV2xSec to secure packets. Therefore, sign and verify of the Csm are being used.

5.5 AUTOSAR NvM

The NvM [7] is used by CnV2xSec to load certificates used for pseudonyms, signature generation and verification of V2X messages.

6 Requirements Tracing

Requirement	Description	Satisfied by
[CP_SRS_CnV2X_-00100]	The implementation of Chinese V2X communication shall follow technical requirements given by CCSA and NTCAS	[CP_SWS_CnV2xSec_00001] [CP_SWS_CnV2xSec_00002] [CP_SWS_CnV2xSec_00003] [CP_SWS_CnV2xSec_00004]
[CP_SRS_CnV2X_-00601]	Digital certificate and secure protocol data unit shall be compliant with CCSA Technical Requirement of Security Certificate Management System for LTE-based Vehicular Communication	[CP_SWS_CnV2xSec_00003] [CP_SWS_CnV2xSec_00004] [CP_SWS_CnV2xSec_00005] [CP_SWS_CnV2xSec_00006] [CP_SWS_CnV2xSec_00007] [CP_SWS_CnV2xSec_01003] [CP_SWS_CnV2xSec_01004] [CP_SWS_CnV2xSec_01005] [CP_SWS_CnV2xSec_02007] [CP_SWS_CnV2xSec_02022] [CP_SWS_CnV2xSec_02025]
[CP_SRS_CnV2X_-00602]	The Chinese V2X communication shall use SM2 elliptic curve public key algorithm and SM3 cryptographic hash algorithm to generate a signature for each BSMs and attach it to the BSM	[CP_SWS_CnV2xSec_00005] [CP_SWS_CnV2xSec_00006] [CP_SWS_CnV2xSec_00007] [CP_SWS_CnV2xSec_00008] [CP_SWS_CnV2xSec_00009] [CP_SWS_CnV2xSec_00010] [CP_SWS_CnV2xSec_02012] [CP_SWS_CnV2xSec_02025]
[CP_SRS_CnV2X_-00603]	The Chinese V2X Communication shall attach a certificate or certificate digest to every BSM and shall not attach CA certificate of certificate chain	[CP_SWS_CnV2xSec_00005]
[CP_SRS_CnV2X_-00604]	The Chinese V2X communication shall not transmit BSMs when it has no valid certificates	[CP_SWS_CnV2xSec_00005]
[CP_SRS_CnV2X_-00605]	The Chinese V2X communication shall randomize the identifiers related to BSM to in order to support privacy	[CP_SWS_CnV2xSec_00011] [CP_SWS_CnV2xSec_02025]
[CP_SRS_CnV2X_-00606]	The Chinese V2X communication shall change pseudonym certificates in order to support privacy	[CP_SWS_CnV2xSec_00011] [CP_SWS_CnV2xSec_00012] [CP_SWS_CnV2xSec_00013] [CP_SWS_CnV2xSec_00014] [CP_SWS_CnV2xSec_00015] [CP_SWS_CnV2xSec_00016] [CP_SWS_CnV2xSec_00017] [CP_SWS_CnV2xSec_02025]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[CP_SWS_CnV2xSec_08001]
[SRS_BSW_00457]	Callback functions of Application software components shall be invoked by the Basis SW	[CP_SWS_CnV2xSec_02021]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 Startup Behavior

[CP_SWS_CnV2xSec_00001]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00100

⌈The function CnV2xSec_Init (refer to chapter 8.3.1) of the CnV2xSec shall initialize the internal states of the CnV2xSec module.⌋

[CP_SWS_CnV2xSec_00002]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00100

⌈The function CnV2xSec_Init shall initialize the basic services of security (refer to chapter 7.2).⌋

7.2 Security Functional Specification

The CnV2xSec module operates the basic services of security.

[CP_SWS_CnV2xSec_00003]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00100, CP_SRS_CnV2X_00601

⌈The CnV2xSec module shall implement the Security Service following technical requirements specified in [2][3].⌋

[CP_SWS_CnV2xSec_00004]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00100, CP_SRS_CnV2X_00601

⌈The CnV2xSec module shall provide the Encap and Decap services required by CnV2xMsg and conduct verification by utilizing Csm.⌋

7.2.1 Encapsulation and Decapsulation

[CP_SWS_CnV2xSec_00005]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00601, CP_SRS_CnV2X_00602, CP_SRS_CnV2X_-00603, CP_SRS_CnV2X_00604

⌈The function CnV2xSec_ReqEncap shall encapsulate the BSM packet to be sent as defined in chapter 6.4 of [12].⌋

[CP_SWS_CnV2xSec_00006]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00601, CP_SRS_CnV2X_00602

⌈The function CnV2xSec_ReqEncap shall generate the SPDU which includes the V2X message, the digital signature of the V2X message, and pseudonym specified in in chapter 6.4 of [13].⌋

[CP_SWS_CnV2xSec_00007]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00601, CP_SRS_CnV2X_00602

⌈The digital signature shall be generated based on SM2 specified in [15].⌋

[CP_SWS_CnV2xSec_00008]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00602

⌈The function CnV2xSec_ReqEncap shall invoke CSM APIs Csm_Hash and Csm_SignatureGenerate for the generation of the digital signature specified in [13].⌋

[CP_SWS_CnV2xSec_00009]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00602

⌈The function CnV2xSec_ReqDecap shall decapsulate the received SPDU. The CnV2xSec module shall verify the pseudonym and digital siginaure in the SPDU specified in chapter 6.5 of [14].⌋

[CP_SWS_CnV2xSec_00010]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00602

⌈The function CnV2xSec_ReqDecap shall invoke Csm APIs Csm_Hash and Csm_SignatureVerify for the verification of the data given in the argument SecuredDataPtr of the function CnV2xSec_ReqDecap.⌋

7.2.2 Pseudonym Management

[CP_SWS_CnV2xSec_00011]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00605, CP_SRS_CnV2X_00606

⌈The pseudonym certificate shall be updated to protect the privacy of the user once the following condition is met: the pseudonym certificate is used for more than 300 seconds, except that the distance is less than 2.1 km or at least one vehicleEventFlags is in the set state.⌋

[CP_SWS_CnV2xSec_00012]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00606

⌈The CnV2xSec shall get time via a call to CnV2xMsg_GetRefTimePtr.⌋

[CP_SWS_CnV2xSec_00013]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00606

⌈The CnV2xSec shall get distance via a call to CnV2xMsg_CheckDistance.⌋

[CP_SWS_CnV2xSec_00014]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00606

⌈The CnV2xSec shall get vehicleEventFlags via a call to CnV2xMsg_GetVehicleEventFlagsStatus.⌋

[CP_SWS_CnV2xSec_00015]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00606

⌈The CnV2xSec shall inform CnV2xMsg to change the pseudonym certificate via a call to CnV2xMsg_PreparePseudonymChange when the condition to change the pseudonym certificate is met.⌋

[CP_SWS_CnV2xSec_00016]

Status:	DRAFT
Upstream requirements:	CP_SRS_CnV2X_00606

⌈The CnV2xSec shall inform CnV2xMsg to chang the pseudonym certificate via a call to CnV2xMsg_CommitPseudonymChange when all modules are ready for the pseudonym certificate change.⌋

[CP_SWS_CnV2xSec_00017]

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00606

⌈The CnV2xSec shall inform CnV2xMsg to change the pseudonym certificate via a call to CnV2xMsg_AbortPseudonymChange when not all modules are ready for the pseudonym certificate change and the change is to be rolled back.⌋

7.3 Error Classification

7.3.1 Development Errors

[SWS_CnV2xSec_00501] Definiton of development errors in module CnV2xSec

Status: DRAFT

⌈

Type of error	*Related error code*	*Error value*
API service called with wrong parameter	CNV2XSEC_E_PARAM	0x01
API service called with invalid pointer	CNV2XSEC_E_PARAM_POINTER	0x02
API function called before the CnV2xSec module has been fully initialized	CNV2XSEC_E_UNINIT	0x03
CnV2xSec initialization failed	CNV2XSEC_E_INIT_FAILED	0x04

⌋

7.3.2 Runtime Errors

There is no runtime errors.

7.3.3 Production Errors

There is no production errors.

7.3.4 Extended Production Errors

There is no extended production errors.

8 API specification

8.1 Imported types

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

[CP_SWS_CnV2xSec_01001] Definition of imported datatypes of module CnV2x Sec

Status: DRAFT

⌈

Module	*Header File*	*Imported Type*
Csm	Rte_Csm_Type.h	Crypto_OperationModeType
	Rte_Csm_Type.h	Crypto_VerifyResultType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
	Rte_NvM_Type.h	NvM_RequestResultType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋

8.2 Type definitions

8.2.1 CnV2xSec_ConfigType

[CP_SWS_CnV2xSec_01002] Definition of datatype CnV2xSec_ConfigType

Status: DRAFT

⌈

Name	CnV2xSec_ConfigType (draft)	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	The content of the configuration data structure is implementation specific.
Description	Configuration data structure of the CnV2xSec module. **Tags:** atp.Status=draft	
Available via	CnV2xSec.h	

⌋

8.2.2 CnV2xSec_SecProfileType

[CP_SWS_CnV2xSec_01003] Definition of datatype CnV2xSec_SecProfileType

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00601

⌈

Name	CnV2xSec_SecProfileType (draft)		
Kind	Enumeration		
Range	CNV2XSEC_SECPROF_ BSM_SIGNED	–	Security Profile for BSM message that are signed only
	CNV2XSEC_SECPROF_ BSM_SIGNED_ DEFLECTED_ENCRYPTED	–	Security Profile for BSM message that are position deflected (using ordinary plug-in), encrypted and signed.
	CNV2XSEC_SECPROF_ BSM_SIGNED_ HIGHDEFLECTED_ ENCRYPTED	–	Security Profile for BSM message that are position deflected (using other plug-in), encrypted and signed.
	CNV2XSEC_SECPROF_ OTHER_SIGNED	–	Security Profile for other message types that have to be signed
Description	Used to describe the security service invoked by V2xSec **Tags:** atp.Status=draft		
Available via	CnV2x_Sec.h		

⌋

8.2.3 CnV2xSec_SecReportType

[CP_SWS_CnV2xSec_01004] Definition of datatype CnV2xSec_SecReportType

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00601

⌈

Name	CnV2xSec_SecReportType (draft)		
Kind	Type		
Derived from	uint8		
Range	CNV2XSEC_SECREP_ SUCCESS	0x00	Indicating security service has successfully executed
	CNV2XSEC_SECREP_ FALSE_SIGNATURE	0x01	Indicating false signature
	CNV2XSEC_SECREP_ INVALID_CERTIFICATE	0x02	Indicating invalid certificate
	CNV2XSEC_SECREP_ REVOKED_CERTIFICATE	0x03	Indicating revoked certificate

▽

△

	CNV2XSEC_SECREP_ INCONSISTENT_	0x04	Indicating inconsistent certificate chain
	CNV2XSEC_SECREP_ TIME_EXPIRED	0x05	Indicating time expired
	CNV2XSEC_SECREP_ DUPLICATE_MESSAGE	0x06	Indicating duplicate message
	CNV2XSEC_SECREP_ INVALID_MOBILITY_DATA	0x07	Indicating invalid mobility data
	CNV2XSEC_SECREP_ UNSIGNED_MESSAGE	0x08	Indicating unsigned message
	CNV2XSEC_SECREP_ SIGNER_CERTIFICATE_ NOT_FOUND	0x09	Indicating signer certificate not found
	CNV2XSEC_SECREP_ UNSUPPORTED_SIGNER_ IDENTIFIER_TYPE	0x0a	Indicating unsupported signer identifier type
	CNV2XSEC_SECREP_ INCOMPATIBLE_ PROTOCOL	0x0b	Indicating incompatible protocol version
	CNV2XSEC_SECREP_ UNENCRYPTED_ MESSAGE	0x0c	Indicating unencrypted message
	CNV2XSEC_SECREP_ DECRYPTION_ERROR	0x0d	Indicating decryption error
	CNV2XSEC_SECREP_ UNSUPPORTED_ SIGNATURE_ALGORITHM	0x0e	Indicating unsupported signature algorithm
	CNV2XSEC_ SECREP_ AID_MISMATCH	0x0f	Indicating mismatch between AID in Secured protocol data Unit(SPDU) and AID in Pseudonym Certifate
	CNV2XSEC_SECREP_ UNFINISH	0xfd	Indicating security execution is unfinished
	CNV2XSEC_ SECREP_ ERROR_OTHER	0xfe	Indicating security service error caused by other reasons
	CNV2XSEC_SECREP_ NONE	0xff	Indicating no security service has been executed
Description	Used to describe the security report after invocation of security services for Decapsulation (verify or decrypt) **Tags:** atp.Status=draft		
Available via	CnV2x_GeneralTypes.h		

⌋

8.2.4 CnV2xSec_SecReturnType

[CP_SWS_CnV2xSec_01005] Definition of datatype CnV2xSec_SecReturnType

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00601

⌈

Name	CnV2xSec_SecReturnType (draft)		
Kind	Enumeration		
Range	CNV2XSEC_E_OK	0x00	Return with success
	CNV2XSEC_E_NOT_OK	0x01	Failure during operation
	CNV2XSEC_E_BUF_OVFL	0x02	Destination buffer too small for security operation data output
Description	Used to return values of security related functions **Tags:** atp.Status=draft		
Available via	CnV2x_Sec.h		

⌋

8.3 Function definitions

8.3.1 CnV2xSec_Init

[CP_SWS_CnV2xSec_02001] Definition of API function CnV2xSec_Init

Status: DRAFT

⌈

Service Name	CnV2xSec_Init (draft)	

Syntax	`void CnV2xSec_Init (`
	`  const CnV2xSec_ConfigType* CfgPtr`
	`)`	

Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	CfgPtr	ConfigPtr Pointer to the selected configuration set
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Initialize the CnV2xSec module **Tags:** atp.Status=draft	
Available via	CnV2xSec.h	

⌋

[CP_SWS_CnV2xSec_02002]

Status: DRAFT

⌈The function CnV2xSec_Init shall store the access to the configuration structure for subsequent API calls.⌋

[CP_SWS_CnV2xSec_02003]

Status: DRAFT

⌈If development error detection is enabled: The function CnV2xSec_Init shall check the parameter CfgPtr for containing a valid configuration. If the check fails, the function CnV2xSec_Init shall raise the development error CNV2XSEC_E_INIT_FAILED.⌋

8.3.2 CnV2xSec_GetVersionInfo

[CP_SWS_CnV2xSec_02005] Definition of API function CnV2xSec_GetVersionInfo

Status: DRAFT

⌈

Service Name	CnV2xSec_GetVersionInfo (draft)	

Syntax	`void CnV2xSec_GetVersionInfo (`
	`  Std_VersionInfoType* VersionInfoPtr`
	`)`	

Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	VersionInfoPtr	Pointer to where to store the version information of this module.
Return value	None	
Description	Returns the version information of this module. **Tags:** atp.Status=draft	
Available via	CnV2xSec.h	

⌋

[CP_SWS_CnV2xSec_02006]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_GetVersionInfo shall check the parameter VersionInfoPtr for being valid. If the check fails, the function CnV2xSec_GetVersionInfo shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

8.3.3 CnV2xSec_ReqEncap

[CP_SWS_CnV2xSec_02007] Definition of API function CnV2xSec_ReqEncap

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00601

⌈

Service Name	CnV2xSec_ReqEncap (draft)	
Syntax	`CnV2xSec_SecReturnType CnV2xSec_ReqEncap ( uint16 TransactionId16, CnV2xSec_SecProfileType SecProfile, uint16 UnsecuredDataLength, const uint8* UnsecuredDataPtr, uint16* SecuredDataLength, uint8* SecuredDataPtr )`	
Service ID [hex]	0x03	
Sync/Async	Synchronous or Async, depending on the job configuration	
Reentrancy	Non Reentrant	
Parameters (in)	TransactionId16	The request identifier that the client can use to match the response
	SecProfile	The security profile to use for encapsulation
	UnsecuredDataLength	The length of the data to use for encapsulation
	UnsecuredDataPtr	The pointer to the data to use for encapsulation
Parameters (inout)	SecuredDataLength	The length pointer containing the maximum length of secured data SecuredDataPtr at input direction. Shall contain the actual size of the secured data SecuredDataPtr at output direction.
	SecuredDataPtr	The pointer where the secured data shall be put.
Parameters (out)	None	
Return value	CnV2xSec_SecReturnType	CnV2XSEC_E_OK: request successful CnV2XSEC_E_NOT_OK: request failed CnV2XSEC_E_BUF_OVFL: SecuredDataLength is too small for security operation result data
Description	This function is called by the CnV2xMsg to generate the SPDU, which includes the V2X message, the signature and pseudonym. An asynchronous CnV2xMsg_EncapConfirmation call will be used to notify CnV2xMsg of the result. **Tags:** atp.Status=draft	
Available via	CnV2xSec.h	

⌋

[CP_SWS_CnV2xSec_02008]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqEncap shall check that the service CnV2xSec_Init was previously called. If the check fails, the function CnV2xSec_ReqEncap shall raise the development error CNV2XSEC_E_UNINIT.⌋

[CP_SWS_CnV2xSec_02009]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqEncap shall check the parameter UnsecuredDataPtr for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqEncap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

[CP_SWS_CnV2xSec_02010]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqEncap shall check the parameter SecuredDataLength for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqEncap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

[CP_SWS_CnV2xSec_02011]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqEncap shall check the parameter SecuredDataPtr for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqEncap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

8.3.4 CnV2xSec_ReqDecap

[CP_SWS_CnV2xSec_02012] Definition of API function CnV2xSec_ReqDecap

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00602

⌈

Service Name	CnV2xSec_ReqDecap (draft)	

Syntax	`CnV2xSec_SecReturnType CnV2xSec_ReqDecap (`
	`  uint32 TransactionId32,`
	`  uint16 SecuredDataLength,`
	`  const uint8* SecuredDataPtr,`
	`  uint16* UnsecuredDataLength,`
	`  uint8* UnsecuredDataPtr,`
	`  CnV2xSec_SecReportType* SecReport,`
	`  uint32* Aid`
	`)`	

Service ID [hex]	0x04	
Sync/Async	Synchronous or Async, depending on the job configuration	
Reentrancy	Non Reentrant	
Parameters (in)	TransactionId32	Transaction Id of the received Packet

▽

△

	SecuredDataLength	The length of the data to decrypt and verify
	SecuredDataPtr	The pointer to the data to decrypt and verify
Parameters (inout)	UnsecuredDataLength	The pointer to the data length of the unsecured data.Shall contain the aximum available length (incoming direction) and the actual used length (outgoing direction)
Parameters (out)	UnsecuredDataPtr	The pointer where the decrypted /verified data shall be put
	SecReport	The security report.
	Aid	The numerical value of the AID
Return value	CnV2xSec_SecReturn Type	CnV2XSEC_E_OK: request successful CnV2XSEC_E_NOT_OK: request failed CnV2XSEC_E_BUF_OVFL: UnsecuredDataLength is too small for security operation result data
Description	This function is called by the CnV2xMsg to decapsulate the SPDU. An asynchronous CnV2x Msg_DecapConfirmation call will be used to notify CnV2xMsg of the result. **Tags:** atp.Status=draft	
Available via	CnV2xSec.h	

⌋

[CP_SWS_CnV2xSec_02013]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqDecap shall check that the service CnV2xSec_Init was previously called. If the check fails, the function CnV2xSec_ReqDecap shall raise the development error CNV2XSEC_E_UNINIT.⌋

[CP_SWS_CnV2xSec_02014]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqDecap shall check the parameter SecuredDataPtr for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqDecap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

[CP_SWS_CnV2xSec_02015]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqDecap shall check the parameter UnsecuredDataLength for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqDecap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

[CP_SWS_CnV2xSec_02016]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqDecap shall check the parameter UnsecuredDataPtr for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqDecap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

[CP_SWS_CnV2xSec_02017]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqDecap shall check the parameter SecReport for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqDecap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

[CP_SWS_CnV2xSec_02018]

Status: DRAFT

⌈To process the Csm_SignatureVerify, CsmSignatureVerifyAlgorithmFamily is set to CRYPTO_ALGOFAM_SM2, CsmSignatureVerifyAlgorithmSecondaryFamily is set to CRYPTO_ALGOFAM_SM3, and the operation mode is a disjunction of the 3 modes START, UPDATE, FINISH.⌋

[CP_SWS_CnV2xSec_02019]

Status: DRAFT

⌈If development error detection is enabled: the function CnV2xSec_ReqDecap shall check the parameter Aid for being valid. If the check fails (i.e., it is a NULL pointer), the function CnV2xSec_ReqDecap shall raise the development error CNV2XSEC_E_PARAM_POINTER.⌋

8.4 Call-back notifications

8.4.1 CSM callback interfaces

[CP_SWS_CnV2xSec_02021]

Status: DRAFT
Upstream requirements: SRS_BSW_00457

⌈If the CnV2xSec module uses the Csm module asynchronously to calculate or verify the signatures, CnV2xSec shall provide callback functions according to Csm_CallbackType.⌋

8.5 Scheduled functions

8.5.1 CnV2xSec_MainFunction

[CP_SWS_CnV2xSec_02022] Definition of scheduled function CnV2xSec_MainFunction

Status: DRAFT
Upstream requirements: CP_SRS_CnV2X_00601

⌈

Service Name	CnV2xSec_MainFunction (draft)
Syntax	`void CnV2xSec_MainFunction ( void )`
Service ID [hex]	0x05
Description	Scheduled MainFunction of CnV2xSec **Tags:** atp.Status=draft
Available via	SchM_CnV2xSec.h

⌋

[CP_SWS_CnV2xSec_02023]

Status: DRAFT

⌈The main function is used for cyclic pseudonym change.⌋

8.6 Expected interfaces

In this chapter, all external interfaces required from other modules are listed.

8.6.1 Mandatory interfaces

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

[CP_SWS_CnV2xSec_02025] Definition of mandatory interfaces required by module CnV2xSec

Status: DRAFT

Upstream requirements: CP_SRS_CnV2X_00601, CP_SRS_CnV2X_00602, CP_SRS_CnV2X_-00605, CP_SRS_CnV2X_00606

⌈

API Function	***Header File***	***Description***
Csm_Hash	Csm.h	Uses the given data to perform the hash calculation and stores the hash.
Csm_SignatureGenerate	Csm.h	Uses the given data to perform the signature calculation and stores the signature in the memory location pointed by the result pointer.
Csm_SignatureVerify	Csm.h	Verifies the given signature by checking if it was generated with the given data.
NvM_GetErrorStatus	NvM.h	Service to read the block dependent error/status information.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.

⌋

8.6.2 Optional Interfaces

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

[CP_SWS_CnV2xSec_02026] Definition of optional interfaces requested by module CnV2xSec

Status: DRAFT

⌈

API Function	***Header File***	***Description***
Det_ReportError	Det.h	Service to report development errors.

⌋

9 Sequence diagrams

9.1 Encapsulation

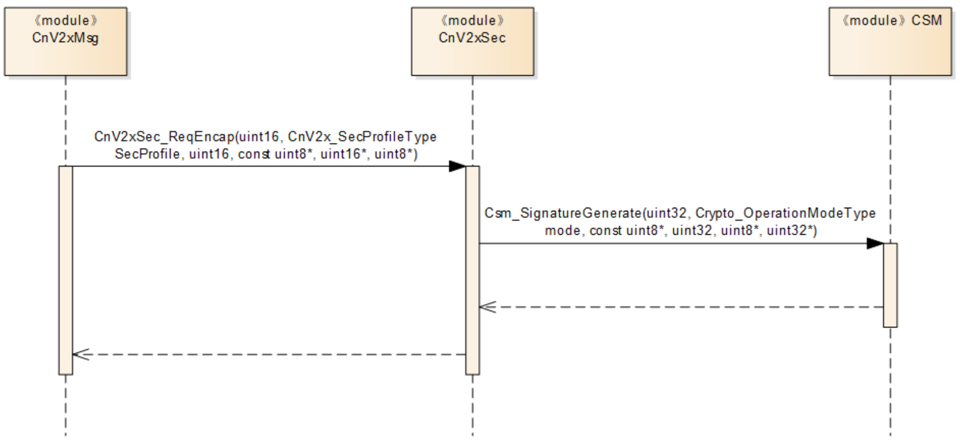
«module» CnV2xMsg
«module» CnV2xSec
«module» CSM
CnV2xSec_ReqEncap(uint16, CnV2x_SecProfileType SecProfile, uint16, const uint8*, uint16*, uint8*)
Csm_SignatureGenerate(uint32, Crypto_OperationModeType mode, const uint8*, uint32, uint8*, uint32*)

Figure 9.1: Encapsulation for Synchronous

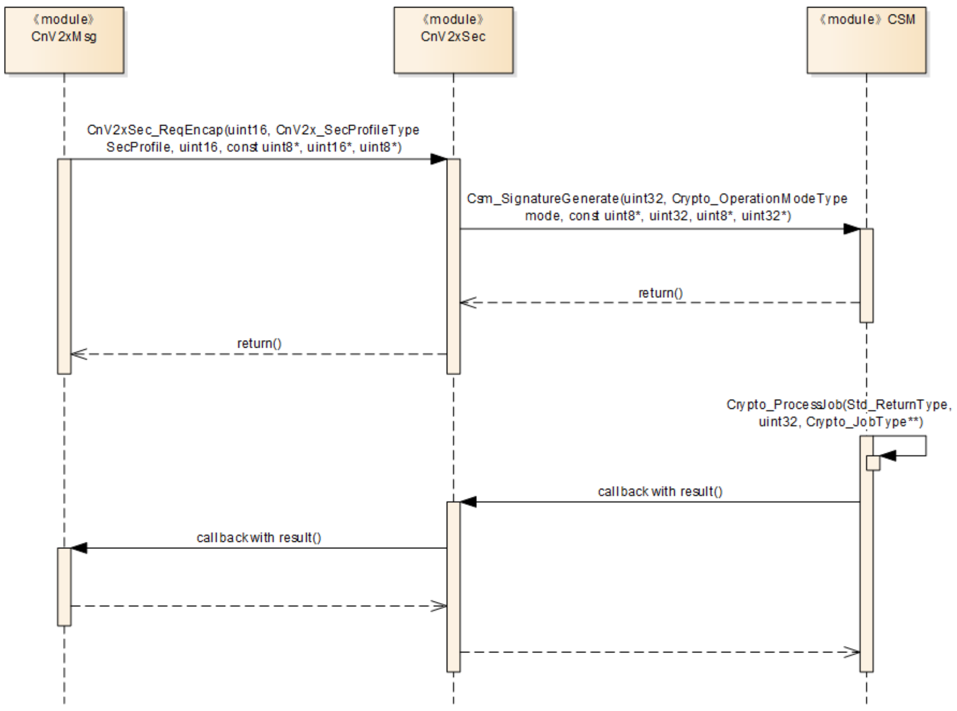
«module» CnV2xMsg
«module» CnV2xSec
«module» CSM
CnV2xSec_ReqEncap(uint16, CnV2x_SecProfileType SecProfile, uint16, const uint8*, uint16*, uint8*)
Csm_SignatureGenerate(uint32, Crypto_OperationModeType mode, const uint8*, uint32, uint8*, uint32*)
return()
return()
Crypto_ProcessJob(Std_ReturnType, uint32, Crypto_JobType**)
callback with result()
callbackwith result()

Figure 9.2: Encapsulation for Asynchronous

9.2 Decapsulation

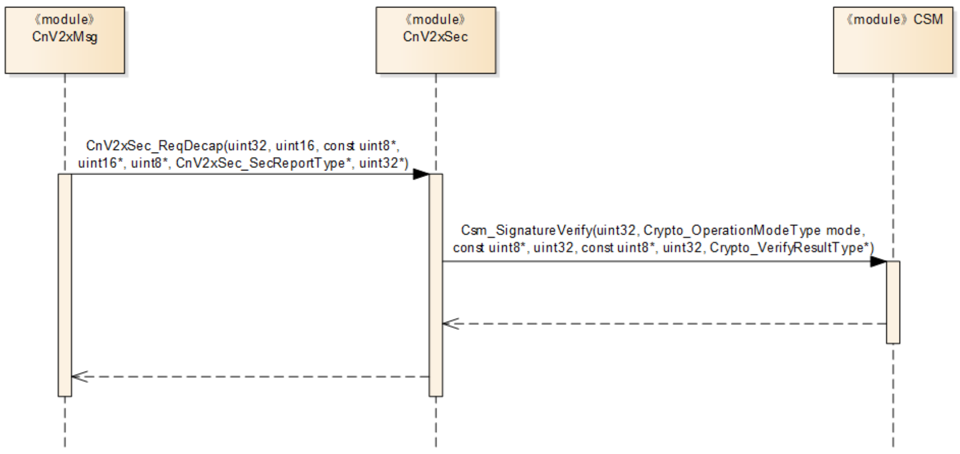
«module» CnV2xMsg
«module» CnV2xSec
«module» CSM
CnV2xSec_ReqDecap(uint32, uint16, const uint8*, uint16*, uint8*, CnV2xSec_SecReportType*, uint32*)
Csm_SignatureVerify(uint32, Crypto_OperationModeType mode, const uint8*, uint32, const uint8*, uint32, Crypto_VerifyResultType*)

Figure 9.3: Decapsulation for Synchronous

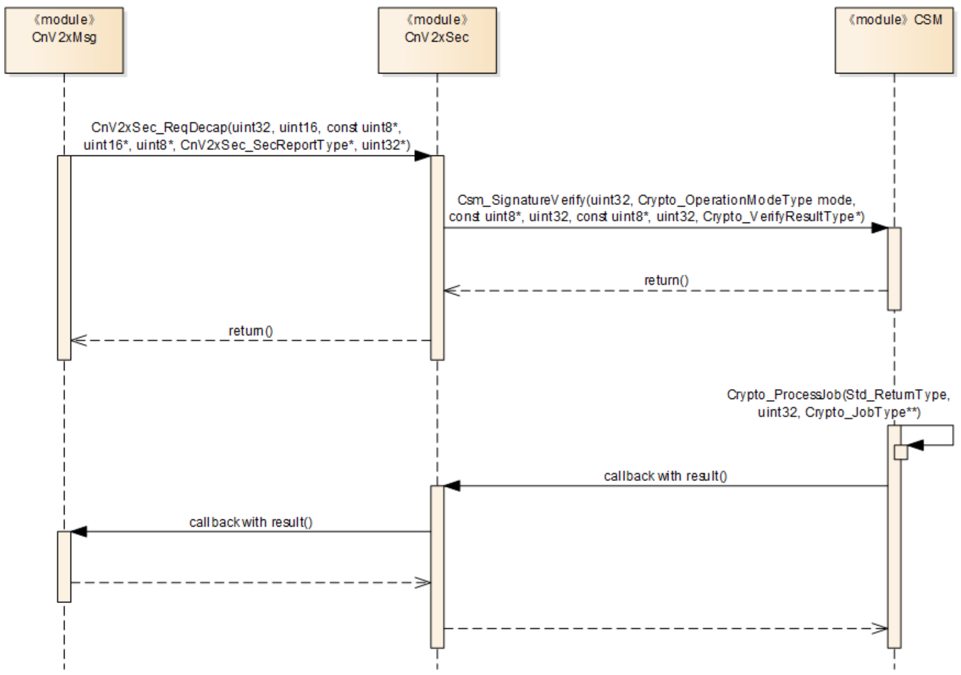
«module» CnV2xMsg
«module» CnV2xSec
«module» CSM
CnV2xSec_ReqDecap(uint32, uint16, const uint8*, uint16*, uint8*, CnV2xSec_SecReportType*, uint32*)
Csm_SignatureVerify(uint32, Crypto_OperationModeType mode, const uint8*, uint32, const uint8*, uint32, Crypto_VerifyResultType*)
return()
return()
Crypto_ProcessJob(Std_ReturnType, uint32, Crypto_JobType**)
callback with result()
callback with result()

Figure 9.4: Decapsulation for Asynchronous

9.3 Update Pseudonym

Please see the sequence diagrams in [6].

10 Configuration specification

10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in Chapter 7 and Chapter 8.

10.1.1 Variants

[CP_SWS_CnV2xSec_08001]
Status: DRAFT
Upstream requirements: SRS_BSW_00345

⌈The CnV2xSec module only supports VARIANT-PRE-COMPILE.⌋

10.1.2 CnV2xSec

[ECUC_CnV2xSec_00001] Definition of EcucModuleDef CnV2xSec
Status: DRAFT

⌈

Module Name	CnV2xSec
Description	Configuration of the CnV2xSec module.
Post-Build Variant Support	false
Supported Config Variants	VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
CnV2xSecGeneral	1	This container contains the configuration parameters of the security module CnV2xSec. **Tags:** atp.Status=draft

⌋

10.1.3 CnV2xSecGeneral

[ECUC_CnV2xSec_00002] Definition of EcucParamConfContainerDef CnV2xSec General

Status: DRAFT

⌈

Container Name	CnV2xSecGeneral
Parent Container	CnV2xSec
Description	This container contains the configuration parameters of the security module CnV2xSec. **Tags:** atp.Status=draft
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
CnV2xSecHashConfigRef	1	[ECUC_CnV2xSec_00006]
CnV2xSecNvMBlockDescriptorLongTermCertificateRef	1	[ECUC_CnV2xSec_00003]
CnV2xSecNvMBlockDescriptorPseudonymCertificateRef	1	[ECUC_CnV2xSec_00004]
CnV2xSecNvMBlockDescriptorRef	1	[ECUC_CnV2xSec_00005]
CnV2xSecSignatureGenerationConfigRef	1	[ECUC_CnV2xSec_00007]
CnV2xSecSignatureVerifyConfigRef	1	[ECUC_CnV2xSec_00008]

No Included Containers

⌋

[ECUC_CnV2xSec_00006] Definition of EcucReferenceDef CnV2xSecHashConfigRef

Status: DRAFT

⌈

Parameter Name	CnV2xSecHashConfigRef		
Parent Container	CnV2xSecGeneral		
Description	Select Csm service configuration that is used for hash. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CnV2xSec_00003] Definition of EcucReferenceDef CnV2xSecNvMBlockDescriptorLongTermCertificateRef

Status: DRAFT

⌈

Parameter Name	CnV2xSecNvMBlockDescriptorLongTermCertificateRef		
Parent Container	CnV2xSecGeneral		
Description	Reference to NVRAM block containing the none volatile data of long term certificates **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to NvMBlockDescriptor		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CnV2xSec_00004] Definition of EcucReferenceDef CnV2xSecNvMBlockDescriptorPseudonymCertificateRef

Status: DRAFT

⌈

Parameter Name	CnV2xSecNvMBlockDescriptorPseudonymCertificateRef		
Parent Container	CnV2xSecGeneral		
Description	Reference to NVRAM block containing the none volatile data of pseudonym certificates. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to NvMBlockDescriptor		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CnV2xSec_00005] Definition of EcucReferenceDef CnV2xSecNvMBlockDescriptorRef

Status: DRAFT

⌈

Parameter Name	CnV2xSecNvMBlockDescriptorRef		
Parent Container	CnV2xSecGeneral		
Description	Reference to NVRAM block containing the none volatile data. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to NvMBlockDescriptor		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CnV2xSec_00007] Definition of EcucReferenceDef CnV2xSecSignatureGenerationConfigRef

Status: DRAFT

⌈

Parameter Name	CnV2xSecSignatureGenerationConfigRef		
Parent Container	CnV2xSecGeneral		
Description	Select Csm service configuration that is used for signature generation. **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

[ECUC_CnV2xSec_00008] Definition of EcucReferenceDef CnV2xSecSignatureVerifyConfigRef

Status: DRAFT

⌈

Parameter Name	CnV2xSecSignatureVerifyConfigRef		
Parent Container	CnV2xSecGeneral		
Description	Select Csm service configuration that is used for signature verification **Tags:** atp.Status=draft		
Multiplicity	1		
Type	Symbolic name reference to CsmJob		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

⌋

A Not applicable requirements

No content

B History of Specification Items

Please note that the lists in this chapter also include specification items that have been removed from the specification in a later version. These specification items do not appear as hyperlinks in the document.

B.1 Specification Item History of this document compared to AUTOSAR R23-11.

B.1.1 Added Specification Items in R24-11

none

B.1.2 Changed Specification Items in R24-11

Number	Heading
[CP_SWS_-CnV2xSec_02018]	
[CP_SWS_-CnV2xSec_02025]	Definition of mandatory interfaces required by module CnV2xSec

Table B.1: Changed Specification Items in R24-11

B.1.3 Deleted Specification Items in R24-11

none

B.2 Specification Item History of this document compared to AUTOSAR R22-11.

B.2.1 Added Specification Items in R23-11

none

B.2.2 Changed Specification Items in R23-11

Number	Heading
[CP_SWS_-CnV2xSec_01001]	Definition of imported datatypes of module CnV2xSec
[CP_SWS_-CnV2xSec_01002]	Definition of datatype CnV2xSec_ConfigType
[CP_SWS_-CnV2xSec_01004]	Definition of datatype CnV2xSec_SecReportType
[CP_SWS_-CnV2xSec_02007]	Definition of API function CnV2xSec_ReqEncap
[CP_SWS_-CnV2xSec_02012]	Definition of API function CnV2xSec_ReqDecap
[CP_SWS_-CnV2xSec_02025]	Definition of mandatory interfaces in module CnV2xSec
[CP_SWS_-CnV2xSec_02026]	Definition of optional interfaces in module CnV2xSec

Table B.2: Changed Specification Items in R23-11

B.2.3 Deleted Specification Items in R23-11

none