Document Title	Specification of Communication Stack Types
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	50

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	• Added ListElemStructType • Added TimeTupleType, TimeStampType, TimeStampQualType • Changed the size of PNCHandleType
2022-11-24	R22-11	AUTOSAR Release Management	• Removed reference to CompilerAbstraction
2021-11-25	R21-11	AUTOSAR Release Management	• Added CbkHandleIdType in Type definitions
2020-11-30	R20-11	AUTOSAR Release Management	• Removed IcomConfigIdType and IcomSwitch_ErrorType from Type definitions
2019-11-28	R19-11	AUTOSAR Release Management	• Renamed of general types headers • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Removed Type BusTrcvErrorType because it is not used at all • Updated PduInfoType for addressing in Upper Layers using MetaData • Update of SWS document as per BSW General document
2015-07-31	4.2.2	AUTOSAR Release Management	• Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• MetaData information is added in PduInfoType
2014-03-31	4.1.3	AUTOSAR Release Management	• Added support for Pretended network data type
2013-10-31	4.1.2	AUTOSAR Release Management	• Removed the published information • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Added support for Partial network data type • Revised Notfication type and RetryInfo type • Additional input (SWS_BSW_General) added for SWS_CommunicationStackTypes
2011-12-22	4.0.3	AUTOSAR Administration	• ComStack Artifacts have been generated from BSW Model • Update of SWS document for new traceability mechanism
2010-09-30	3.1.5	AUTOSAR Administration	• Add TPParameterType and Enumeration value TP_NORETRY in RetryInfoType • ComStack_Types.h divided into ComStack_Types.h and ComStack_Cfg.h • PduIdType and PduLengthType defined in ComStack_Cfg.h file

▽

△

2010-02-02	3.1.4	AUTOSAR Administration	• Typo errors are corrected throughout the document • General return codes for NotifResultType has been added to support Tp_ChangeParameterRequest • TpDataStateType and RetryInfoType has been added to store the Tp buffer status information • Common Published information has been updated • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-07-24	2.1.16	AUTOSAR Administration	• Chapter numbers in chapter 8.1 corrected • New data type NetworkHandleType created according item Comtype026 established • Syntax correction in PduInfoType • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added • Changed "sender" to "receiver" at NTFRSLT_E_WRT_OVRN
2006-11-28	2.1.2	AUTOSAR Administration	• NTFRSLT_E_TIMEOUT_Bs changed NTFRSLT_E_TIMEOUT_BS • NTFRSLT_E_TIMEOUT_Cr changed to NTFRSLT_E_TIMEOUT_CR • Definitions according to compiler abstraction added • Legal disclaimer revised
2006-11-28	2.1.1	AUTOSAR Administration	• Initial release (The V1.0.0 was only as Pre-Release availabel within Release 1.0)

Disclaimer

Contents

1 Introduction and functional overview

This document specifies the AUTOSAR communication stack type header file. It contains all types that are used across several modules of the communication stack of the basic software and all types of all basic software modules that are platform and compiler independent.

It is strongly recommended that those communication stack type files are unique within the AUTOSAR community to guarantee unique types and to avoid type changes when changing from supplier A to B.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the Communication Stack Types that are not included in the [1, AUTOSAR glossary].

Acronym:	Description:
API	Application Programming Interface
DCM	Diagnostic Communication Manager
I-PDU	Interaction Layer PDU. In AUTOSAR the Interaction Layer is equivalent to the Communication Services Layer.
L-PDU	Data Link Layer PDU. In AUTOSAR the Data Link Layer is equivalent to the Communication Hardware Abstraction and Microcontroller Abstraction Layer.
N-PDU	Network Layer PDU. In AUTOSAR the Network Layer is equivalent to the Transport Protocol.
OSEK/VDX	In May 1993 OSEK has been founded as a joint project in the German automotive industry aiming at an industry standard for an open-ended architecture for distributed control units in vehicles. OSEK is an abbreviation for the German term "Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug" (English: Open Systems and the Corresponding Interfaces for Automotive Electronics). Initial project partners were BMW, Bosch, Daimler Chrysler, Opel, Siemens, VW and the IIIT of the University of Karlsruhe as co-ordinator. The French car manufacturers PSA and Renault joined OSEK in 1994 introducing their VDX-approach (Vehicle Distributed eXecutive) which is a similar project within the French automotive industry. At the first workshop on October 1995 the OSEK/VDX group presented the results of the harmonised specification between OSEK and VDX. After the 2nd international OSEK/VDX Workshop in October 1997 the 2nd versions of the specifications were published.
PDU	Protocol Data Unit
SDU	Service Data Unit - Payload of PDU
TP	Transport Protocol

Table 2.1: Acronyms used in the scope of this Document

Abbreviation:	Description:
Com	Communication
EcuC	ECU Configuration
e.g.	[lat.] exempli gratia = [eng.] for example
i.e.	[lat.] it est = [eng.] that is

Table 2.2: Abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] Requirements on Communication
AUTOSAR_CP_RS_COM

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for Communication Stack Types.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Communication Stack Types.

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

4.3 Applicability to safety related environments

No restrictions, because the subject of this specification is a header file specifying types. It does not include or implement any functionality.

5 Dependencies to other modules

The communication stack type header file defines communication types based on the platform types [PltfTypes] (Platform_Types.h) header file. To prevent multiple includes of header files, the communication stack header file includes the standard types header file [StdTypes] which already includes both other files.

6 Requirements Tracing

The following tables reference the requirements specified in [3] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_COMTYPE_91005]
[SRS_Com_02043]	AUTOSAR COM and LargeDataCOM shall provide a receive indication function	[SWS_Comtype_00004] [SWS_Comtype_00006] [SWS_Comtype_00007] [SWS_Comtype_00010] [SWS_Comtype_00014] [SWS_Comtype_00015] [SWS_Comtype_00017] [SWS_Comtype_00030]
[SRS_Com_02045]	AUTOSAR COM and LargeDataCOM shall provide a function to request the transmit buffer data for lower layer triggered transmission	[SWS_Comtype_00004] [SWS_Comtype_00006] [SWS_Comtype_00007] [SWS_Comtype_00010] [SWS_Comtype_00014] [SWS_Comtype_00015] [SWS_Comtype_00017] [SWS_Comtype_00030]
[SRS_Com_02095]	AUTOSAR COM and LargeDataCOM shall use the TP to fragment and reassemble large signals	[SWS_Comtype_00004] [SWS_Comtype_00006] [SWS_Comtype_00007] [SWS_Comtype_00010] [SWS_Comtype_00014] [SWS_Comtype_00015] [SWS_Comtype_00017] [SWS_Comtype_00030]
[SRS_Com_02114]	AUTOSAR COM and LargeDataCOM shall support independent development of CP Software Clusters	[SWS_COMTYPE_91001]
[SRS_Eth_00105]	Support of time stamping in hardware	[SWS_COMTYPE_91002] [SWS_COMTYPE_91003] [SWS_COMTYPE_91004]
[SRS_Eth_00167]	PTP Physical Clock Adjustment	[SWS_COMTYPE_91002]
[SRS_Eth_00172]	Ethernet Driver hardware supported data transfer	[SWS_COMTYPE_91005]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General issues

[SWS_Comtype_00004]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈It is not allowed to add any project or supplier specific extension to this file. Any extension invalidates the AUTOSAR conformity.⌋

[SWS_Comtype_00015]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈Because many of the communication stack type are depending on the appropriate ECU, this file shall be generated dependent on the specific ECU configuration for each ECU independently.⌋

[SWS_Comtype_00030]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈The value of `PduIdType` and `PduLengthType` shall be derived from the 'PduIdType Enum' and 'PduLengthTypeEnum' of the EcuCPduCollection container respectively.⌋

7.2 Error Classification

Section "Error Handling" of the document [2] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.2.1 Development Errors

There are no development errors.

7.2.2 Runtime Errors

There are no runtime errors.

7.2.3 Production Errors

There are no production errors.

7.2.4 Extended Production Errors

There are no extended production errors.

7.3 Security Events

The module does not report security events.

8 API specification

8.1 Type definitions

8.1.1 PduIdType

[SWS_COMTYPE_00005] Definition of datatype PduIdType ⌈

Name	PduIdType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	The size of this global type depends on the maximum number of PDUs used within one software module.	
	uint8	The size of this global type depends on the maximum number of PDUs used within one software module.	
Range	0...<PduIdmax>	–	Zero-based integer number The size of this global type depends on the maximum number of PDUs used within one software module. This parameter shall be generated by the generator tool depending on the value configured in EcuC virtual layer. This parameter shall be generated in ComStack_Cfg.h file Example : If **no** software module deals with more PDUs that 256, this type can be set to uint8. If at least one software module handles more than 256 PDUs, this type must globally be set to uint16.
Description	This type is used within the entire AUTOSAR Com Stack except for bus drivers.		
Available via	ComStack_Types.h		

⌋

[SWS_Comtype_00006]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈Variables of this type serve as a unique identifier of a PDU within a software module or a set thereof, and also for interaction of two software modules where the PduId of the corresponding target module is being used for referencing.⌋

[SWS_Comtype_00007]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈In order to be able to perform table-indexing within a software module, variables of this type shall be zero-based and consecutive.

There might be several ranges of PduIds in a module, one for each type of operation performed within that module (e.g. sending and receiving).⌋

[SWS_Comtype_00014]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈PduIdmax, the maximum number of a PduId range, is the number -1 of PDUs dealt with in the corresponding type of operation within that module.⌋

8.1.2 PduLengthType

[SWS_COMTYPE_00008] Definition of datatype PduLengthType ⌈

Name	PduLengthType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	The size of this global type depends on the maximum length of PDUs to be sent by an ECU.	
	uint32	The size of this global type depends on the maximum length of PDUs to be sent by an ECU.	
	uint8	The size of this global type depends on the maximum length of PDUs to be sent by an ECU.	
Range	0...<PduLengthmax>	–	Zero-based integer number The size of this global type depends on the maximum length of PDUs to be sent by an ECU. This parameter shall be generated by the generator tool depending on the value configured in EcuC virtual layer. This parameter shall be generated in ComStack_Cfg.h file Example : If no segmentation is used the length depends on the maximum payload size of a frame of the underlying communication system (for FlexRay maximum size is 255, therefore uint8). If segmentation is used it depends on the maximum length of a segmented N-PDU (in general uint16 is used)
Description	This type shall be used within the entire AUTOSAR Com Stack of an ECU except for bus drivers.		
Available via	ComStack_Types.h		

⌋

[SWS_Comtype_00010]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈Variables of this type serve as length information of a PDU. The length information is provided in number of bytes.⌋

[SWS_Comtype_00017]

Upstream requirements: SRS_Com_02043, SRS_Com_02045, SRS_Com_02095

⌈PduLengthmax, the maximum length of a Pdu, is the length of the largest (possibly segmented) PDU to be sent by the ECU.⌋

8.1.3 PduInfoType

[SWS_COMTYPE_00011] Definition of datatype PduInfoType ⌈

Name	PduInfoType	
Kind	Structure	
Elements	SduDataPtr	
	Type	uint8*
	Comment	Pointer to the SDU (i.e. payload data) of the PDU. The type of this pointer depends on the memory model being used at compile time.
	MetaDataPtr	
	Type	uint8*
	Comment	Pointer to the meta data (e.g. CAN ID, socket ID, diagnostic addresses) of the PDU, consisting of a sequence of meta data items. The length and type of the meta data items is statically configured for each PDU. Meta data items with more than 8 bits use platform byte order.
	SduLength	
	Type	PduLengthType
	Comment	Length of the SDU in bytes.
Description	Variables of this type shall be used to store the basic information about a PDU of any type, namely a pointer variable pointing to its SDU (payload), a pointer to Meta Data of the PDU, and the corresponding length of the SDU in bytes.	
Available via	ComStack_Types.h	

⌋

8.1.4 PNCHandleType

[SWS_COMTYPE_00036] Definition of datatype PNCHandleType ⌈

Name	PNCHandleType
Kind	Type
Derived from	uint16
Description	Used to store the identifier of a partial network cluster.
Available via	ComStack_Types.h

⌋

8.1.5 TPParameterType

[SWS_COMTYPE_00031] Definition of datatype TPParameterType ⌈

Name	TPParameterType		
Kind	Enumeration		
Range	TP_STMIN	0x00	Separation Time
	TP_BS	0x01	Block Size
	TP_BC	0x02	The Band width control parameter used in FlexRay transport protocol module.
Description	Specify the parameter to which the value has to be changed (BS or STmin).		
Available via	ComStack_Types.h		

⌋

8.1.6 BufReq_ReturnType

[SWS_COMTYPE_00012] Definition of datatype BufReq_ReturnType ⌈

Name	BufReq_ReturnType		
Kind	Enumeration		
Range	BUFREQ_OK	0x00	Buffer request accomplished successful. This status shall have the value 0.
	BUFREQ_E_NOT_OK	0x01	Buffer request not successful. Buffer cannot be accessed. This status shall have the value 1.
	BUFREQ_E_BUSY	0x02	Temporarily no buffer available. It's up the requester to retry request for a certain time. This status shall have the value 2.
	BUFREQ_E_OVFL	0x03	No Buffer of the required length can be provided. This status shall have the value 3.
Description	Variables of this type shall be used to store the result of a buffer request.		
Available via	ComStack_Types.h		

⌋

8.1.7 TpDataStateType

[SWS_COMTYPE_00027] Definition of datatype TpDataStateType ⌈

Name	TpDataStateType		
Kind	Enumeration		
Range	TP_DATACONF	0x00	TP_DATACONF indicates that all data, that have been copied so far, are confirmed and can be removed from the TP buffer. Data copied by this API call are excluded and will be confirmed later.
	TP_DATARETRY	0x01	TP_DATARETRY indicates that this API call shall copyalready copied data in order to recover from an error. In this case TxTpData Cnt specifies the offset of the first byte to be copied by the API call.
	TP_CONFPENDING	0x02	TP_CONFPENDING indicates that the previously copied data must remain in the TP.
Description	Variables of this type shall be used to store the state of TP buffer.		
Available via	ComStack_Types.h		

⌋

8.1.8 RetryInfoType

[SWS_COMTYPE_00037] Definition of datatype RetryInfoType ⌈

Name	RetryInfoType	
Kind	Structure	
Elements	TpDataState	
	Type	TpDataStateType
	Comment	The enum type to be used to store the state of Tp buffer.
	TxTpDataCnt	
	Type	PduLengthType
	Comment	Offset from the current position which identifies the number of bytes to be retransmitted.
Description	Variables of this type shall be used to store the information about Tp buffer handling.	
Available via	ComStack_Types.h	

⌋

8.1.9 NetworkHandleType

[SWS_COMTYPE_00038] Definition of ImplementationDataType NetworkHandleType ⌈

Name	NetworkHandleType		
Kind	Type		
Derived from	uint8		
Range	0..255	–	Zero-based integer number
Description	Variables of the type NetworkHandleType shall be used to store the identifier of a communication channel.		
Variation	–		
Available via	ComStack_Types.h		

⌋

8.1.10 CbkHandleIdType

[SWS_COMTYPE_91001] Definition of datatype CbkHandleIdType

Status: DRAFT
Upstream requirements: SRS_Com_02114

⌈

Name	CbkHandleIdType (draft)
Kind	Type
Derived from	uint16
Description	Used for the handle Ids of Com and LdCom user callbacks. **Tags:** atp.Status=draft
Available via	ComStack_Types.h

⌋

8.1.11 TimeTupleType

[SWS_COMTYPE_91002] Definition of datatype TimeTupleType

Status: DRAFT
Upstream requirements: SRS_Eth_00167, SRS_Eth_00105

⌈

Name	TimeTupleType (draft)	
Kind	Structure	
Elements	timestampClockValue	
	Type	TimeStampType
	Comment	Value of the clock, which is used of ingress/egress timestamping
	disciplinedClockValue	
	Type	TimeStampType
	Comment	Value of the adjustable HW clock
	timeQuality	
	Type	TimeStampQualType
	Comment	Status of time tuple
Description	The Time Tuple represents the clock values of two related HW clocks • the value of the clock used for timestamping of frames • and the corresponding value of the adjustable HW clock, derived by cross-timestamping **Tags:** atp.Status=draft	
Available via	ComStackTypes.h	

⌋

8.1.12 TimeStampType

[SWS_COMTYPE_91003] Definition of datatype TimeStampType

Status: DRAFT
Upstream requirements: SRS_Eth_00105

⌈

Name	TimeStampType (draft)	
Kind	Structure	
Elements	nanoseconds	
	Type	uint32
	Comment	Nanoseconds part of the time
	seconds	
	Type	uint32
	Comment	32 bit LSB of the 48 bits Seconds part of the time

▽

△

	secondsHi	
	Type	uint16
	Comment	16 bit MSB of the 48 bits Seconds part of the time
Description	Variables of this type are used for expressing time stamps including relative time and absolute calendar time. The absolute time starts at 1970-01-01. Value range of Seconds part: • 0 .. (2 ^{48} -1), i.e. 0 to 3257812230d [0xFFFF FFFF FFFF] Value range of Nanoseconds part: • 0 to 999999999ns [0x3B9AC9FF] • invalid value in nanoseconds: [0x3B9ACA00] to [0x3FFFFFFF] • Bit 30 and 31 reserved, default: 0 **Tags:** atp.Status=draft	
Available via	ComStackTypes.h	

⌋

8.1.13 TimeStampQualType

[SWS_COMTYPE_91004] Definition of datatype TimeStampQualType

Status: DRAFT
Upstream requirements: SRS_Eth_00105

⌈

Name	TimeStampQualType (draft)		
Kind	Enumeration		
Range	VALID	0	Timestamp is valid
	INVALID	1	Timestamp is invalid
	UNCERTAIN	2	Status of timestamp is uncertain
Description	Depending on the HW, quality information regarding the evaluated timestamp might be supported. If not supported, the value shall be always Valid. For Uncertain and Invalid values, the upper layer shall discard the time stamp. **Tags:** atp.Status=draft		
Available via	ComStackTypes.h		

⌋

8.1.14 ListElemStructType

[SWS_COMTYPE_91005] Definition of datatype ListElemStructType

Status: DRAFT
Upstream requirements: SRS_BSW_00441, SRS_Eth_00172

⌈

Name	ListElemStructType (draft)	
Kind	Structure	
Elements	DataLength	
	Type	uint16
	Comment	Represents length of data
	DataPtr	
	Type	uint8*
	Comment	Represents pointer to data
	NextListElemPtr	
	Type	ListElemStructType*
	Comment	Pointer to next list element
Description	This type defines one element of a single linked list. Each element represents on part of an associated data block. The data block could form for example an Ethernet frame. Each element addresses a data location, the data length and a pointer to the next element. The last node (tail) has NextListElemPtr set to NUL_PTR. The re-construction process of a data block (e.g. Ethernet frame) is performed by traversing from the first data element (head) to the last data element (tail). The single linked list is linked in network order (big-endian). Thus, the head element represents the most significiant part of the data block (e.g. Ethernet frame) **Tags:** atp.Status=draft	
Available via	ComStack_Types.h	

⌋

8.2 Function definitions

Not applicable.

9 Sequence diagrams

Not applicable.

10 Configuration specification

10.1 Published Information

For details refer to the chapter 10.3 “Published Information” in [2].

A Not applicable requirements

[SWS_Comtype_NA]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00101, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00171, SRS_BSW_00323, SRS_BSW_00336, SRS_BSW_00337, SRS_BSW_00339, SRS_BSW_00344, SRS_BSW_00345, SRS_BSW_00369, SRS_BSW_00380, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00402, SRS_BSW_00403, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00406, SRS_BSW_00407, SRS_BSW_00409, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00450, SRS_BSW_00451, SRS_BSW_00452, SRS_BSW_00458, SRS_BSW_00461, SRS_BSW_00466, SRS_BSW_00467, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00488, SRS_BSW_00489, SRS_BSW_00490, SRS_BSW_00491, SRS_BSW_00492, SRS_BSW_00493, SRS_Com_00177, SRS_Com_00192, SRS_Com_00218, SRS_Com_02030, SRS_Com_02037, SRS_Com_02040, SRS_Com_02041, SRS_Com_02042, SRS_Com_02044, SRS_Com_02046, SRS_Com_02058, SRS_Com_02067, SRS_Com_02077, SRS_Com_02078, SRS_Com_02079, SRS_Com_02080, SRS_Com_02082, SRS_Com_02083, SRS_Com_02084, SRS_Com_02086, SRS_Com_02087, SRS_Com_02088, SRS_Com_02089, SRS_Com_02090, SRS_Com_02091, SRS_Com_02092, SRS_Com_02093, SRS_Com_02094, SRS_Com_02096, SRS_Com_02097, SRS_Com_02098, SRS_Com_02107, SRS_Com_02108, SRS_Com_02109, SRS_Com_02110, SRS_Com_02111, SRS_Com_02112, SRS_Com_02113

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R24-11

B.1.1 Added Specification Items in R24-11

none

B.1.2 Changed Specification Items in R24-11

none

B.1.3 Deleted Specification Items in R24-11

none