Document Title	Specification of Standard Types
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	49

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	• Editorial Changes
2023-11-23	R23-11	AUTOSAR Release Management	• Added Safety Transformer Error Codes in [SWS_Std_00028] • Editorial Changes
2022-11-24	R22-11	AUTOSAR Release Management	• Added Std_TransformerForwardCode and removed [SWS_Std_00027] • Introduced Not applicable requirement [SWS_Std_NA_00999] • Removed use of deprecate "compiler abstraction" from [SWS_Std_00031] • Editorial Changes
2021-11-25	R21-11	AUTOSAR Release Management	• Added SWS_Std_00031 (NULL_PTR) • Editorial Changes
2020-11-30	R20-11	AUTOSAR Release Management	• Fixed Design issues with E2E communication protection for methods • Added TransformerError and TransformerForward • Fixed missing Type definitions • Editorial Changes

▽

△

2019-11-28	R19-11	AUTOSAR Release Management	• Added chapter Std_TransformerError • Editorial changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Header File Cleanup (no impact on behavior)
2017-12-08	4.3.1	AUTOSAR Release Management	• Updated OSEK reference (editorial)
2016-11-30	4.3.0	AUTOSAR Release Management	• Corrected editorial traceability issues
2015-07-31	4.2.2	AUTOSAR Release Management	• Harmonized traceability
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Harmonized requirements according to SWS_General
2011-12-22	4.0.3	AUTOSAR Administration	• Update of SWS documents for new traceability mechanism
2010-02-02	3.1.4	AUTOSAR Administration	• Removed instanceID from StdVersionType • Concretized the published parameters to have the prefix STD_TYPES • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Add Module ID for Complex Drivers • Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added

▽

△

2006-11-28-01	2.1.1	AUTOSAR Administration	• Changed definition of Standard_ReturnType to match the RTE definition. • A complete overview of definitions and values has been performed to match the requirements in the SRS General.
2006-05-16	2.0	AUTOSAR Administration	• Initial Release

Disclaimer

Contents

1 Introduction and functional overview

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

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

2 Acronyms and abbreviations

Acronyms and abbreviations that have a local scope are not contained in the AUTOSAR glossary. These must appear in a local glossary.

Acronym:	Description:
API	Application Programming Interface
OSEK/VDX	Offene Systeme und deren Schnittstellen fuer die Elektronik im Kraftfahrzeug

Abreviation:	Description:
STD	Standard

3 Related documentation

3.1 Input documents & related standards and norms

[1] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[2] General Requirements on SPAL
AUTOSAR_CP_RS_SPALGeneral

[3] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[4] Requirements on Basic Software Module Description Template
AUTOSAR_CP_RS_BSWModuleDescriptionTemplate

[5] ISO 17356-3: Road vehicles – Open interface for embedded automotive applications – Part 3: OSEK/VDX Operating System (OS)

[6] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [1, SWS BSW General], which is also valid for Standard Types.

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

Further specification:

[2, SRS SPALGeneral] [3, SWS RTE] [4, RS BSW General] [5, OSEK/VDX Operating System] [ISO/IEC 9899:1990]

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

Many symbols defined in this specification (like OK, NOT_OK, ON, OFF) are already defined and used within legacy software. These conflicts ('redefinition of existing symbol') are expected, but neglected, because of the following reasons:

1. AUTOSAR has to maintain network compatibility with legacy ECUs, but no software architecture compatibility with legacy software Many types are defined and used exactly in the same way that legacy software does. Legacy software can keep on using the symbols, only the definitions have to be removed and taken from this file instead.

5 Software Architecture

5.1 Dependencies to other modules

5.2 File structure

The include structures differ between BSW modules which are part of the COM-stack and other modules. BSW modules which is considered part of the COM stack shall include the ComStackTypes.h other modules shall include StandardTypes.h

5.2.1 Communication related BSW modules

[SWS_Std_00030] ⌈The include file structure shall be as follows:

- ComStackTypes.h shall include StandardTypes.h
- Communication related basic software modules shall include ComStackTypes.h

⌋

6 Requirements Tracing

The following tables reference the requirements specified in SRS BSW General [1] 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_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_Std_00015] [SWS_Std_91003]
[SRS_BSW_00161]	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	[SWS_Std_00004]
[SRS_BSW_00305]	Data types naming convention	[SWS_Std_00017] [SWS_Std_00019] [SWS_Std_91001] [SWS_Std_91002]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_Std_00007] [SWS_Std_00010] [SWS_Std_00013]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_Std_00005] [SWS_Std_00006] [SWS_Std_00011]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_Std_00011]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_Std_00031]
[SRS_Xfrm_00002]	A transformer shall provide fixed interfaces	[SWS_Std_00028] [SWS_Std_00029]
[SRS_Xfrm_00004]	A transformer shall support error handling	[SWS_Std_00021] [SWS_Std_00022] [SWS_Std_00024] [SWS_Std_00025]
[SRS_Xfrm_00008]	A transformer shall specify its output format	[SWS_Std_00022] [SWS_Std_00023]
[SRS_Xfrm_00009]	A fixed set of transformer classes shall exist	[SWS_Std_00023]
[SRS_Xfrm_00010]	Each transformer class shall provide a fixed set of abstract errors	[SWS_Std_00024]
[SRS_Xfrm_00011]	A transformer shall belong to a specific transformer class	[SWS_Std_00026]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 General issues

[SWS_Std_00004]

Upstream requirements: SRS_BSW_00161

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

[SWS_Std_00014] ⌈The standard types header file shall be protected against multiple inclusion:

```
#ifndef STD_TYPES_H

#define STD_TYPES_H

..

/*

* Contents of file

*/

..

#endif /* STD_TYPES_H */
```

⌋

7.2 Error Classification

The section "Error Handling" of the document "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.

8 API specification

8.1 Type definitions

8.1.1 Std_ReturnType

[SWS_Std_00005] Definition of datatype Std_ReturnType

Upstream requirements: SRS_BSW_00357

⌈

Name	Std_ReturnType		
Kind	Type		
Derived from	uint8		
Range	E_OK	0	see 8.2.1, SWS_Std_00006
	E_NOT_OK	1	see 8.2.1, SWS_Std_00006
	0x02-0x3F	2	Available to user specific errors
Description	This type can be used as standard API return type which is shared between the RTE and the BSW modules. It shall be defined as follows: typedef uint8 Std_ReturnType;		
Available via	Std_Types.h		

⌋

[SWS_Std_00011]

Upstream requirements: SRS_BSW_00357, SRS_BSW_00441

⌈The Std_ReturnType shall normally be used with value E_OK or E_NOT_OK. If those return values are not sufficient user specific values can be defined by using the 6 least specific bits.

For the naming of the user defined values the module prefix shall be used as requested in SRS_BSW_00441

Layout of the Std_ReturnType shall be as stated in the RTE specification. Bit 7 and Bit 8 are reserved and defined by the RTE specification.

⌋

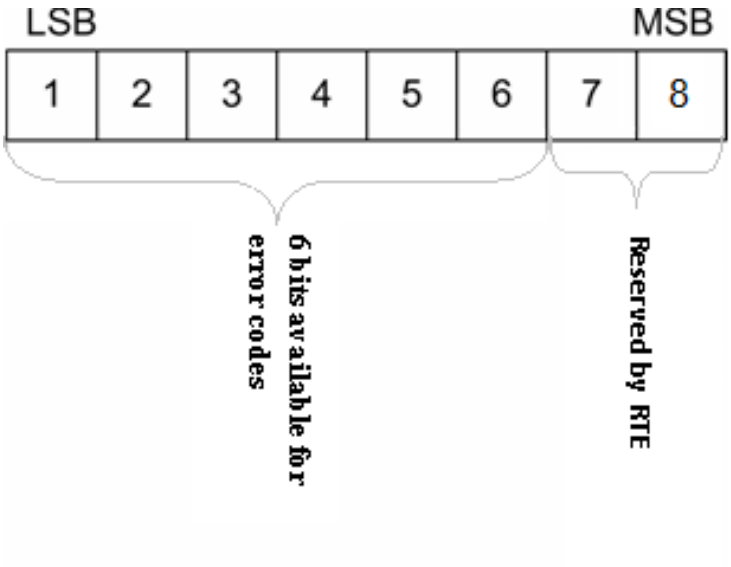

Figure 8.1: Layout of Std_Return_Type

8.1.2 Std_VersionInfoType

[SWS_Std_00015] Definition of datatype Std_VersionInfoType

Upstream requirements: SRS_BSW_00004

⌈

Name	Std_VersionInfoType	
Kind	Structure	
Elements	vendorID	
	Type	uint16
	Comment	–
	moduleID	
	Type	uint16
	Comment	–
	sw_major_version	
	Type	uint8
	Comment	–
	sw_minor_version	
	Type	uint8
	Comment	–
	sw_patch_version	
	Type	uint8
	Comment	–
Description	This type shall be used to request the version of a BSW module using the <Module name>_Get VersionInfo() function.	
Available via	Std_Types.h	

⌋

8.1.3 Std_TransformerError

The data type Std_TransformerError is a struct which contains the error code and the transformer class to which the error belongs.

The data type Std_TransformerError shall be defined as follows:

[SWS_Std_00021] Definition of datatype Std_TransformerError

Upstream requirements: SRS_Xfrm_00004

⌈

Name	Std_TransformerError	
Kind	Structure	
Elements	errorCode	
	Type	Std_TransformerErrorCode
	Comment	The specific meaning of the values of Std_TransformerErrorCode is to be seen for the specific transformer chain for which the data type represents the transformer error.
	transformerClass	
	Type	Std_TransformerClass
	Comment	–
Description	Std_TransformerError represents a transformer error in the context of a certain transformer chain.	
Available via	Std_Types.h	

⌋

The values are specified for each transformer class in [26, ASWS Transformer General].

[SWS_Std_00022] Definition of datatype Std_TransformerErrorCode

Upstream requirements: SRS_Xfrm_00004, SRS_Xfrm_00008

⌈

Name	Std_TransformerErrorCode		
Kind	Type		
Derived from	uint8		
Range	-	–	The values are specified for each transformer class in ASWS_ TransformerGeneral.
Description	The type of the Std_TransformerError.		
Available via	Std_Types.h		

⌋

The Std_TransformerClass represents the transformer class in which the error occurred.

[SWS_Std_00023]

Upstream requirements: SRS_Xfrm_00009, SRS_Xfrm_00008

⌈The underlying data type of the type Std_TransformerClass shall be uint8.⌋

The type Std_TransformerClass shall be an enumeration with the following elements where each element represents a transformer class:

[SWS_Std_00024] Definition of datatype Std_TransformerClass

Upstream requirements: SRS_Xfrm_00004, SRS_Xfrm_00010

⌈

Name	Std_TransformerClass		
Kind	Type		
Derived from	uint8		
Range	STD_TRANSFORMER_UNSPECIFIED	0x00	Transformer of a unspecified transformer class.
	STD_TRANSFORMER_SERIALIZER	0x01	Transformer of a serializer class.
	STD_TRANSFORMER_SAFETY	0x02	Transformer of a safety class.
	STD_TRANSFORMER_SECURITY	0x03	Transformer of a security class.
	STD_TRANSFORMER_CUSTOM	0xFF	Transformer of a custom class not standardized by AUTOSAR.
Description	Std_TransformerClass is an enumeration where each element represents a transformer class.		
Available via	Std_Types.h		

⌋

[SWS_Std_00025]

Upstream requirements: SRS_Xfrm_00004

⌈The transformer class STD_TRANSFORMER_UNSPECIFIED shall be used if no transformer error occured.⌋

[SWS_Std_00026]

Upstream requirements: SRS_Xfrm_00011

⌈The mapping from transformerClass of TransformationTechnology to value of data type Std_TransformerClass shall be:

- transformerClass serializer - STD_TRANSFORMER_SERIALIZER
- transformerClass safety - STD_TRANSFORMER_SAFETY
- transformerClass security - STD_TRANSFORMER_SECURITY
- transformerClass custom - STD_TRANSFORMER_CUSTOM

⌋

8.1.4 Std_TransformerForwardCode

The data type Std_TransformerForwardCode represents a forwarded transformer code in the context of a certain transformer chain (see [6]).

The specific meaning of the values of Std_TransformerForwardCode is always to be seen for the specific transformer chain for which the data type represents the transformer status.

[SWS_Std_00028] Safety Transformer Error Codes

Status: DRAFT
Upstream requirements: SRS_Xfrm_00002

⌈

Error Name	Error Code	Description
E_OK	0x00	No specific error to be injected
E_SAFETY_INVALID_REP	0x01	Repeat the last used sequence number.
E_SAFETY_INVALID_CRC	0x03	Generate a deliberately wrong CRC.
E_SAFETY_INVALID_SEQ	0x02	Use a wrong sequence number.

⌋

The underlying data type of the type Std_TransformerForwardCode shall be uint8:

[SWS_Std_00029] Definition of datatype Std_TransformerForwardCode

Status: DRAFT
Upstream requirements: SRS_Xfrm_00002

⌈

Name	Std_TransformerForwardCode (draft)		
Kind	Type		
Derived from	uint8		
Range	E_OK	0x00	–
	E_SAFETY_INVALID_REP	0x01	–
	E_SAFETY_INVALID_SEQ	0x02	–
	E_SAFETY_INVALID_CRC	0x03	–
Description	– **Tags:** atp.Status=draft		
Available via	Std_Types.h		

⌋

8.1.5 Std_MessageTypeType

[SWS_Std_91001] Definition of datatype Std_MessageTypeType

Upstream requirements: SRS_BSW_00305

⌈

Name	Std_MessageTypeType		
Kind	Type		
Derived from	uint8		
Range	STD_MESSAGETYPE_REQUEST	0x00	Message type for a request message
	STD_MESSAGETYPE_RESPONSE	0x01	Message type for a response message
	0x02-0x3F	0x02	reserverd for future message type
Description	This type is used to encode the different type of messages. - Currently this encoding is limited to the distinction between requests and responses in C/S communication.		
Available via	Std_Types.h		

⌋

[SWS_Std_00017]

Upstream requirements: SRS_BSW_00305

⌈The Std_MessageTypeType shall be used ot encode the different types of messages exchanged in AUTOSAR. - Currently this encoding is limited to the distinction between requests and responses in C/S communication.⌋

Note: In future AUTOSAR release, the literals for this type may be extended with additional message types.

8.1.6 Std_MessageResultType

[SWS_Std_91002] Definition of datatype Std_MessageResultType

Upstream requirements: SRS_BSW_00305

⌈

Name	Std_MessageResultType		
Kind	Type		
Derived from	uint8		
Range	STD_MESSAGERESULT_OK	0x00	STD_MESSAGERESULT_OK
	STD_MESSAGERESULT_ERROR	0x01	Messageresult for an ERROR response

▽

△

	0x02-0x3F	0x02	reserverd for future message results
Description	This type is used to encode different types of results for response messages. - Currently this encoding is limited to the distinction between OK and ERROR responses.		
Available via	Std_Types.h		

⌋

[SWS_Std_00019]

Upstream requirements: SRS_BSW_00305

⌈The Std_MessageResultType shall be used ot encode the different types of results for response messages. - Currently this encoding is limited to the distinction between OK and ERROR responses.⌋

Note: In future AUTOSAR release, the literals for this type may be extended with additional result types.

8.1.7 Std_ExtractProtocolHeaderFieldsType

[SWS_Std_91003] Definition of datatype Std_ExtractProtocolHeaderFieldsType

Upstream requirements: SRS_BSW_00004

⌈

Name	Std_ExtractProtocolHeaderFieldsType	
Kind	Function Pointer	
Syntax	`Std_ReturnType (*Std_ExtractProtocolHeaderFieldsType) ( const uint8* buffer, uint32 bufferLength, Std_MessageTypeType* messageType, Std_MessageResultType* messageResult )`	
Parameters (in)	buffer	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer
	bufferLength	Length of the buffer
Parameters (inout)	None	
Parameters (out)	messageType	Canonical representation of the message type (extracted from the transformers protocol header).
	messageResult	Canonical representation of the message result type (extracted from the transformers protocol header).
Return value	Std_ReturnType	–
Description	Type for the function pointer to extract the relevant protocol header fields of the message and the type of the message result of a transformer. - At the time being, this is limited to the types used for C/S communication (i.e., REQUEST and RESPONSE and OK and ERROR).	
Available via	Std_Types.h	

⌋

8.2 Symbol definitions

8.2.1 E_OK, E_NOT_OK

[SWS_Std_00006] Definition of datatype E_OK, E_NOT_OK

Upstream requirements: SRS_BSW_00357

⌈

Name	E_OK, E_NOT_OK		
Kind	Enumeration		
Range	E_OK	0x00u	–
	E_NOT_OK	0x01u	–
Description	Because E_OK is already defined within OSEK, the symbol E_OK has to be shared. To avoid name clashes and redefinition problems, the symbols have to be defined in the following way (approved within implementation): #ifndef STATUSTYPEDEFINED #define STATUSTYPEDEFINED #define E_OK 0x00u typedef unsigned char StatusType; /* OSEK compliance */ #endif #define E_NOT_OK 0x01u		
Available via	Std_Types.h		

⌋

8.2.2 STD_HIGH, STD_LOW

[SWS_Std_00007] Definition of datatype STD_HIGH, STD_LOW

Upstream requirements: SRS_BSW_00348

⌈

Name	STD_HIGH, STD_LOW		
Kind	Enumeration		
Range	STD_LOW	0x00u	–
	STD_HIGH	0x01u	–
Description	The symbols STD_HIGH and STD_LOW shall be defined as follows: #define STD_HIGH 0x01u /* Physical state 5V or 3.3V */ #define STD_LOW 0x00u /* Physical state 0V */		
Available via	Std_Types.h		

⌋

8.2.3 STD_ACTIVE, STD_IDLE

[SWS_Std_00013] Definition of datatype STD_ACTIVE, STD_IDLE

Upstream requirements: SRS_BSW_00348

⌈

Name	STD_ACTIVE, STD_IDLE		
Kind	Enumeration		
Range	STD_IDLE	0x00u	–
	STD_ACTIVE	0x01u	–
Description	The symbols STD_ACTIVE and STD_IDLE shall be defined as follows: #define STD_ACTIVE 0x01u /* Logical state active */ #define STD_IDLE 0x00u /* Logical state idle */		
Available via	Std_Types.h		

⌋

8.2.4 STD_ON, STD_OFF

[SWS_Std_00010] Definition of datatype STD_ON, STD_OFF

Upstream requirements: SRS_BSW_00348

⌈

Name	STD_ON, STD_OFF		
Kind	Enumeration		
Range	STD_OFF	0x00u	–
	STD_ON	0x01u	–
Description	The symbols STD_ON and STD_OFF shall be defined as follows: #define STD_ON 0x01u #define STD_OFF 0x00u		
Available via	Std_Types.h		

⌋

8.2.5 NULL_PTR

[SWS_Std_00031]

Upstream requirements: SRS_BSW_00480

⌈

Define	NULL_PTR	
Range	void pointer	`((void *)0)`
Description	The implementation shall provide the NULL_PTR define with a void pointer to zero definition.	

⌋

8.3 Function definitions

Not applicable.

9 Sequence diagrams

Not applicable.

10 Configuration specification

Not applicable.

A Not applicable requirements

[SWS_Std_NA_00999]

Upstream requirements: SRS_BSW_00101, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, 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_00375, SRS_BSW_00380, SRS_BSW_00383, 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_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_00452, SRS_BSW_00458, SRS_BSW_00466

⌈These requirements are not applicable to this specification.⌋

B History of Requirements

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

B.1 Requirement History of this Document According to AUTOSAR Release R22-11

B.1.1 Added Specification Items in R22-11

[SWS_Std_NA_00999]

B.1.2 Changed Specification Items in R22-11

[SWS_Std_00005] [SWS_Std_00006] [SWS_Std_00007] [SWS_Std_00010] [SWS_Std_00013] [SWS_Std_00015] [SWS_Std_00021] [SWS_Std_00022] [SWS_Std_00024] [SWS_Std_00029] [SWS_Std_00031] [SWS_Std_91001] [SWS_Std_91002] [SWS_Std_91003]

B.1.3 Deleted Specification Items in R22-11

[SWS_Std_00027] [SWS_Std_00999]

B.2 Requirement History of this Document According to AUTOSAR Release R23-11

B.2.1 Added Specification Items in R23-11

none

B.2.2 Changed Specification Items in R23-11

[SWS_Std_00028]

B.2.3 Deleted Specification Items in R23-11

none

B.3 Requirement History of this Document According to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

none

B.3.3 Deleted Specification Items in R24-11

none