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	R22-11

Document Change History

Date	Release	Changed by	Description
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

Contents

1 Introduction and functional overview 7

2 Acronyms and abbreviations 8

3 Related documentation 9

3.1 Input documents & related standards and norms 9
3.2 Related specification 9

4 Constraints and assumptions 10

4.1 Limitations 10
4.2 Applicability to car domains 10

5 Software Architecture 11

5.1 Dependencies to other modules 11
5.2 File structure 11
5.2.1 Communication related BSW modules 11

6 Requirements Tracing 12

7 Functional specification 20

7.1 General issues 20
7.2 Error Classification 20
7.2.1 Development Errors 20
7.2.2 Runtime Errors 20
7.2.3 Transient Faults 21
7.2.4 Production Errors 21
7.2.5 Extended Production Errors 21

8 API specification 22

8.1 Type definitions 22
8.1.1 Std_ReturnType 22
8.1.2 Std_VersionInfoType 23
8.1.3 Std_TransformerError 23
8.1.4 Std_TransformerForwardCode 25
8.1.5 Std_MessageTypeType 26
8.1.6 Std_MessageResultType 26
8.1.7 Std_ExtractProtocolHeaderFieldsType 27
8.2 Symbol definitions 27
8.2.1 E_OK, E_NOT_OK 27
8.2.2 STD_HIGH, STD_LOW 28
8.2.3 STD_ACTIVE, STD_IDLE 28
8.2.4 STD_ON, STD_OFF 28
8.2.5 NULL_PTR 29
8.3 Function definitions 29

9 Sequence diagrams 30

10 Configuration specification 31

A Not applicable requirements 32

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_SRS_BSWGeneral

[2] General Requirements on SPAL
AUTOSAR_SRS_SPALGeneral

[3] Specification of RTE Software
AUTOSAR_SWS_RTE

[4] Requirements on Basic Software Module Description Template
AUTOSAR_RS_BSWModuleDescriptionTemplate

[5] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

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

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, TR BSW General] [6, 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

⌋*(SRS_BSW_00024)*

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_00005]	Modules of the μC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	[SWS_Std_NA_00999]
[SRS_BSW_00006]	The source code of software modules above the μC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	[SWS_Std_NA_00999]
[SRS_BSW_00007]	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	[SWS_Std_NA_00999]
[SRS_BSW_00009]	All Basic SW Modules shall be documented according to a common standard.	[SWS_Std_NA_00999]
[SRS_BSW_00010]	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	[SWS_Std_NA_00999]
[SRS_BSW_00024]	No description	[SWS_Std_00030]
[SRS_BSW_00059]	No description	[SWS_Std_00014]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Std_NA_00999]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_Std_NA_00999]
[SRS_BSW_00160]	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	[SWS_Std_NA_00999]
[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] [SWS_Std_NA_00999]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_Std_NA_00999]
[SRS_BSW_00164]	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	[SWS_Std_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_Std_NA_00999]
[SRS_BSW_00168]	SW components shall be tested by a function defined in a common API in the Basis-SW	[SWS_Std_NA_00999]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_Std_NA_00999]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Std_NA_00999]
[SRS_BSW_00172]	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	[SWS_Std_NA_00999]
[SRS_BSW_00300]	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	[SWS_Std_NA_00999]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_Std_NA_00999]
[SRS_BSW_00302]	All AUTOSAR Basic Software Modules shall only export information needed by other modules	[SWS_Std_NA_00999]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_Std_NA_00999]
[SRS_BSW_00305]	Data types naming convention	[SWS_Std_00017] [SWS_Std_00019] [SWS_Std_91001] [SWS_Std_91002] [SWS_Std_NA_00999]
[SRS_BSW_00306]	AUTOSAR Basic Software Modules shall be compiler and platform independent	[SWS_Std_NA_00999]
[SRS_BSW_00307]	Global variables naming convention	[SWS_Std_NA_00999]
[SRS_BSW_00308]	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	[SWS_Std_NA_00999]
[SRS_BSW_00309]	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	[SWS_Std_NA_00999]
[SRS_BSW_00310]	API naming convention	[SWS_Std_NA_00999]
[SRS_BSW_00312]	Shared code shall be reentrant	[SWS_Std_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_Std_NA_00999]
[SRS_BSW_00321]	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	[SWS_Std_NA_00999]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_Std_NA_00999]
[SRS_BSW_00325]	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	[SWS_Std_NA_00999]
[SRS_BSW_00327]	Error values naming convention	[SWS_Std_NA_00999]
[SRS_BSW_00330]	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	[SWS_Std_NA_00999]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_Std_NA_00999]
[SRS_BSW_00333]	For each callback function it shall be specified if it is called from interrupt context or not	[SWS_Std_NA_00999]
[SRS_BSW_00334]	All Basic Software Modules shall provide an XML file that contains the meta data	[SWS_Std_NA_00999]
[SRS_BSW_00335]	Status values naming convention	[SWS_Std_NA_00999]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Std_NA_00999]
[SRS_BSW_00337]	Classification of development errors	[SWS_Std_NA_00999]
[SRS_BSW_00339]	Reporting of production relevant error status	[SWS_Std_NA_00999]
[SRS_BSW_00341]	Module documentation shall contains all needed informations	[SWS_Std_NA_00999]
[SRS_BSW_00342]	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	[SWS_Std_NA_00999]
[SRS_BSW_00343]	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	[SWS_Std_NA_00999]
[SRS_BSW_00344]	BSW Modules shall support link-time configuration	[SWS_Std_NA_00999]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_Std_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00346]	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	[SWS_Std_NA_00999]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_Std_NA_00999]
[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_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	[SWS_Std_NA_00999]
[SRS_BSW_00353]	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	[SWS_Std_NA_00999]
[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_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Std_NA_00999]
[SRS_BSW_00359]	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	[SWS_Std_NA_00999]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_Std_NA_00999]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_Std_NA_00999]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_Std_NA_00999]
[SRS_BSW_00374]	All Basic Software Modules shall provide a readable module vendor identification	[SWS_Std_NA_00999]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_Std_NA_00999]
[SRS_BSW_00377]	A Basic Software Module can return a module specific types	[SWS_Std_NA_00999]
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_Std_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00379]	All software modules shall provide a module identifier in the header file and in the module XML description file.	[SWS_Std_NA_00999]
[SRS_BSW_00380]	Configuration parameters being stored in memory shall be placed into separate c-files	[SWS_Std_NA_00999]
[SRS_BSW_00383]	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	[SWS_Std_NA_00999]
[SRS_BSW_00385]	List possible error notifications	[SWS_Std_NA_00999]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_Std_NA_00999]
[SRS_BSW_00388]	Containers shall be used to group configuration parameters that are defined for the same object	[SWS_Std_NA_00999]
[SRS_BSW_00389]	Containers shall have names	[SWS_Std_NA_00999]
[SRS_BSW_00390]	Parameter content shall be unique within the module	[SWS_Std_NA_00999]
[SRS_BSW_00392]	Parameters shall have a type	[SWS_Std_NA_00999]
[SRS_BSW_00393]	Parameters shall have a range	[SWS_Std_NA_00999]
[SRS_BSW_00394]	The Basic Software Module specifications shall specify the scope of the configuration parameters	[SWS_Std_NA_00999]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_Std_NA_00999]
[SRS_BSW_00396]	The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/ container	[SWS_Std_NA_00999]
[SRS_BSW_00397]	The configuration parameters in pre-compile time are fixed before compilation starts	[SWS_Std_NA_00999]
[SRS_BSW_00398]	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	[SWS_Std_NA_00999]
[SRS_BSW_00399]	Parameter-sets shall be located in a separate segment and shall be loaded after the code	[SWS_Std_NA_00999]
[SRS_BSW_00400]	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	[SWS_Std_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00401]	Documentation of multiple instances of configuration parameters shall be available	[SWS_Std_NA_00999]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Std_NA_00999]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_Std_NA_00999]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_Std_NA_00999]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_Std_NA_00999]
[SRS_BSW_00408]	All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule	[SWS_Std_NA_00999]
[SRS_BSW_00409]	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	[SWS_Std_NA_00999]
[SRS_BSW_00410]	Compiler switches shall have defined values	[SWS_Std_NA_00999]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_Std_NA_00999]
[SRS_BSW_00413]	An index-based accessing of the instances of BSW modules shall be done	[SWS_Std_NA_00999]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Std_NA_00999]
[SRS_BSW_00415]	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	[SWS_Std_NA_00999]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_Std_NA_00999]
[SRS_BSW_00417]	Software which is not part of the SW-C shall report error events only after the Dem is fully operational.	[SWS_Std_NA_00999]
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as `const` it should be placed into a separate c-file	[SWS_Std_NA_00999]

Requirement	Description	Satisfied by
[SRS_BSW_00422]	Pre-de-bouncing of error status information is done within the Dem	[SWS_Std_NA_00999]
[SRS_BSW_00423]	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	[SWS_Std_NA_00999]
[SRS_BSW_00424]	BSW module main processing functions shall not be allowed to enter a wait state	[SWS_Std_NA_00999]
[SRS_BSW_00425]	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	[SWS_Std_NA_00999]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_Std_NA_00999]
[SRS_BSW_00427]	ISR functions shall be defined and documented in the BSW module description template	[SWS_Std_NA_00999]
[SRS_BSW_00428]	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	[SWS_Std_NA_00999]
[SRS_BSW_00429]	Access to OS is restricted	[SWS_Std_NA_00999]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_Std_NA_00999]
[SRS_BSW_00433]	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	[SWS_Std_NA_00999]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_Std_00011]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_Std_NA_00999]
[SRS_BSW_00458]	Classification of production errors	[SWS_Std_NA_00999]
[SRS_BSW_00466]	Classification of extended production errors	[SWS_Std_NA_00999]
[SRS_BSW_00473]	Classification of transient faults	[SWS_Std_NA_00999]
[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]

Requirement	Description	Satisfied by
[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]

7 Functional specification

7.1 General issues

[SWS_Std_00004] ⌈It is not allowed to add any project or supplier specific extension to this file. Any extension invalidates the AUTOSAR conformity.⌋*(SRS_BSW_00161)*

[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 */
```

⌋*(SRS_BSW_00059)*

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 Transient Faults

There are no transient faults.

7.2.4 Production Errors

There are no production errors.

7.2.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Type definitions

8.1.1 Std_ReturnType

[SWS_Std_00005] ⌈

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		

⌋*(SRS_BSW_00357)*

[SWS_Std_00011] ⌈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.

⌋*(SRS_BSW_00357, SRS_BSW_00441)*

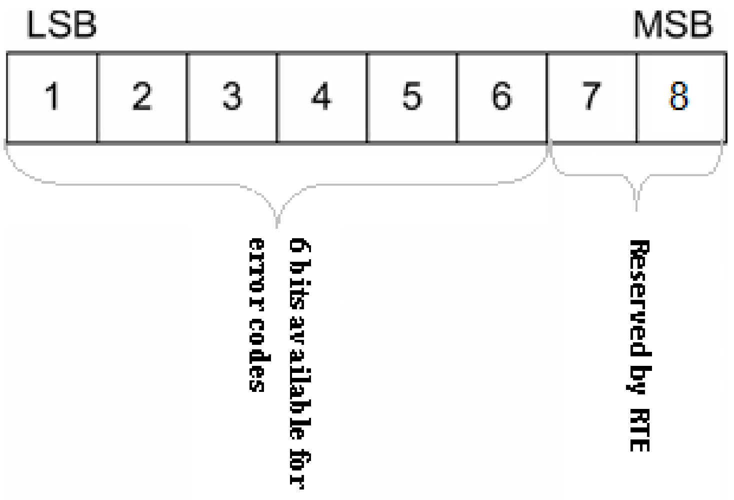

Figure 8.1: Layout of Std_Return_Type

8.1.2 Std_VersionInfoType

[SWS_Std_00015] ⌈

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	

⌋*(SRS_BSW_00004)*

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] ⌈

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	

⌋*(SRS_Xfrm_00004)*

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

[SWS_Std_00022] ⌈

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		

⌋*(SRS_Xfrm_00004, SRS_Xfrm_00008)*

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

[SWS_Std_00023] ⌈The underlying data type of the type Std_TransformerClass shall be uint8.⌋*(SRS_Xfrm_00009, SRS_Xfrm_00008)*

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

[SWS_Std_00024] ⌈

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		

⌋*(SRS_Xfrm_00004, SRS_Xfrm_00010)*

[SWS_Std_00025] ⌈The transformer class STD_TRANSFORMER_UNSPECIFIED shall be used if no transformer error occured.⌋*(SRS_Xfrm_00004)*

[SWS_Std_00026] ⌈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

⌋(*SRS_Xfrm_00011*)

8.1.4 Std_TransformerForwardCode

The data type Std_TransformerForwardCode represents a forwarded transformer code in the context of a certain transformer chain (see [5]).

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]{DRAFT} ⌈

A safety transformer shall handle the forwarded status according to table 8.1.

⌋(*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.

Table 8.1: Safety Transformer Error Codes

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

[SWS_Std_00029]{DRAFT} ⌈

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		

⌋(*SRS_Xfrm_00002*)

8.1.5 Std_MessageTypeType

[SWS_Std_91001] ⌈

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		

⌋*(SRS_BSW_00305)*

[SWS_Std_00017] ⌈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.⌋*(SRS_BSW_-00305)*

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] ⌈

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		

⌋*(SRS_BSW_00305)*

[SWS_Std_00019] ⌈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.⌋*(SRS_BSW_00305)*

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] ⌈

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	

⌋*(SRS_BSW_00004)*

8.2 Symbol definitions

8.2.1 E_OK, E_NOT_OK

[SWS_Std_00006] ⌈

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		

⌋*(SRS_BSW_00357)*

8.2.2 STD_HIGH, STD_LOW

[SWS_Std_00007] ⌈

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		

⌋*(SRS_BSW_00348)*

8.2.3 STD_ACTIVE, STD_IDLE

[SWS_Std_00013] ⌈

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		

⌋*(SRS_BSW_00348)*

8.2.4 STD_ON, STD_OFF

[SWS_Std_00010] ⌈

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		

⌋*(SRS_BSW_00348)*

8.2.5 NULL_PTR

[SWS_Std_00031] ⌈

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

⌋*(SRS_BSW_00480)*

8.3 Function definitions

Not applicable.

9 Sequence diagrams

Not applicable.

10 Configuration specification

Not applicable.

A Not applicable requirements

[SWS_Std_NA_00999] ⌈These requirements are not applicable to this specification.⌋*(SRS_BSW_00300, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00304, SRS_BSW_00305, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00310, SRS_BSW_00312, SRS_BSW_00314, SRS_BSW_00321, SRS_BSW_00325, SRS_BSW_00327, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00341, SRS_BSW_00346, SRS_BSW_00347, SRS_BSW_00350, SRS_BSW_00353, SRS_BSW_00358, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00373, SRS_BSW_00374, SRS_BSW_00377, SRS_BSW_00378, SRS_BSW_00379, SRS_BSW_00401, SRS_BSW_00408, SRS_BSW_00410, SRS_BSW_00411, SRS_BSW_00413, SRS_BSW_00414, SRS_BSW_00415, SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00172, SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00345, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00171, SRS_BSW_00170, SRS_BSW_00380, SRS_BSW_00419, SRS_BSW_00383, 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_00375, SRS_BSW_00101, SRS_BSW_00416, SRS_BSW_00406, SRS_BSW_00168, SRS_BSW_00407, 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_00336, SRS_BSW_00337, SRS_BSW_00369, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00323, SRS_BSW_00409, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00452, SRS_BSW_00473, SRS_BSW_00458, SRS_BSW_00466)*