Document Title	Specification of Communication Stack Types
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	050

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History

Date	Release	Changed by	Change Description
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

Document Change History

Date	Release	Changed by	Change Description
2013-03-15	4.1.1	AUTOSAR Administration	• Added support for Partial network data type • Revised Notification 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

Document Change History

Date	Release	Changed by	Change Description
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_WFT_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 available within Release 1.0)

Disclaimer

Table of 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

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
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, DaimlerChrysler, 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

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

3 Related documentation

3.1 Input documents

[1] [GeneralSRS] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[2] [SRSSPAL] General Requirements on
SPAL AUTOSAR_SRS_SPALGeneral.pdf

[3] [StdTypes] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[4] [PltfTypes] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[5] [CompTypes] Specification of Compiler Abstraction
AUTOSAR_SWS_CompilerAbstraction.pdf

[6] [CANTP] Specification of CAN Transport Layer
AUTOSAR_SWS_CANTransportLayer.pdf

[7] [FlexRayTP] Specification of FlexRay Transport Layer
AUTOSAR_SWS_FlexRayTransportLayer.pdf

[8] [CANTRCV] Specification of CAN Transceiver Driver
AUTOSAR_SWS_CANTransceiverDriver.pdf

[9] [FRTRCV] Specification of FlexRay Transceiver Driver
AUTOSAR_SWS_FlexRayTransceiverDriver.pdf

[10] [BSMDT]Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[11] [BSWModule]List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

[12] [BSWGeneral]General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

[CProgLang] ISO/IEC 9899:1990 Programming Language – C
[ISONM] ISO/IEC 15765-2; 2003 Diagnostics on Controller Area Networks (CAN) – Network layer services

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules (→ chapter 3.1) (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 limitations.

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 Software Architecture

5.1 Dependencies to other modules

The communication stack type header file defines communication types based on the platform types [PltfTypes] (PlatformTypes.h) and Compiler (Compiler.h) header file [CompTypes]. 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 traceability

Requirement	Description	Satisfied by
SRS_BSW_00006	The source code of software modules above the µC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_Comtype_NA_3
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2012 Standard.	SWS_Comtype_NA_3
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_Comtype_NA_3
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_Comtype_NA_3
SRS_BSW_00158	-	SWS_Comtype_NA_0
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_Comtype_NA_0
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_Comtype_NA_0
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_Comtype_NA_0
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_Comtype_NA_1
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_Comtype_NA_1
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_Comtype_NA_0
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_Comtype_NA_3
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_Comtype_NA_3
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_Comtype_NA_3
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Comtype_NA_3
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_Comtype_NA_3
SRS_BSW_00307	Global variables naming convention	SWS_Comtype_NA_3
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_Comtype_NA_3
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_Comtype_NA_3

SRS_BSW_00310	API naming convention	SWS_Comtype_NA_3
SRS_BSW_00312	Shared code shall be reentrant	SWS_Comtype_NA_3
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_Comtype_NA_3
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_Comtype_NA_3
SRS_BSW_00327	Error values naming convention	SWS_Comtype_NA_5
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_Comtype_NA_3
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_Comtype_NA_3
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_Comtype_NA_3
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_Comtype_NA_3
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_Comtype_NA_3
SRS_BSW_00335	Status values naming convention	SWS_Comtype_NA_3
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_Comtype_NA_4
SRS_BSW_00337	Classification of development errors	SWS_Comtype_NA_5
SRS_BSW_00339	Reporting of production relevant error status	SWS_Comtype_NA_5
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_Comtype_NA_3
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_Comtype_NA_0
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_Comtype_NA_0
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_Comtype_NA_0
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_Comtype_NA_0
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_Comtype_NA_0
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_Comtype_NA_3
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall allow the enabling/disabling of detection and reporting of development errors.	SWS_Comtype_NA_5
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_Comtype_NA_3
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_Comtype_NA_3
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_Comtype_NA_3
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_Comtype_NA_3

SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_Comtype_NA_3
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_Comtype_NA_5
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_Comtype_NA_3
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_Comtype_NA_3
SRS_BSW_00374	All Basic Software Modules shall provide a readable module vendor identification	SWS_Comtype_NA_3
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_Comtype_NA_3
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Comtype_NA_3
SRS_BSW_00379	All software modules shall provide a module identifier in the header file and in the module XML description file.	SWS_Comtype_NA_3
SRS_BSW_00380	Configuration parameters being stored in memory shall be placed into separate c-files	SWS_Comtype_NA_0
SRS_BSW_00381	-	SWS_Comtype_NA_0
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_Comtype_NA_0
SRS_BSW_00385	List possible error notifications	SWS_Comtype_NA_5
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_Comtype_NA_5
SRS_BSW_00388	Containers shall be used to group configuration parameters that are defined for the same object	SWS_Comtype_NA_0
SRS_BSW_00389	Containers shall have names	SWS_Comtype_NA_0
SRS_BSW_00390	Parameter content shall be unique within the module	SWS_Comtype_NA_0
SRS_BSW_00392	Parameters shall have a type	SWS_Comtype_NA_0
SRS_BSW_00393	Parameters shall have a range	SWS_Comtype_NA_0
SRS_BSW_00394	The Basic Software Module specifications shall specify the scope of the configuration parameters	SWS_Comtype_NA_0
SRS_BSW_00395	The Basic Software Module specifications shall list all configuration parameter dependencies	SWS_Comtype_NA_0
SRS_BSW_00396	The Basic Software Module specifications shall specify the supported configuration classes for changing values and multiplicities for each parameter/container	SWS_Comtype_NA_0
SRS_BSW_00397	The configuration parameters in pre-compile time are fixed before compilation starts	SWS_Comtype_NA_0
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_Comtype_NA_0
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_Comtype_NA_0
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_Comtype_NA_0

SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_Comtype_NA_0
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_Comtype_NA_0
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_Comtype_NA_0
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_Comtype_NA_3
SRS_BSW_00408	All AUTOSAR Basic Software Modules configuration parameters shall be named according to a specific naming rule	SWS_Comtype_NA_0
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_Comtype_NA_5
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_Comtype_NA_3
SRS_BSW_00412	-	SWS_Comtype_NA_0
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_Comtype_NA_3
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_Comtype_NA_3
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_Comtype_NA_5
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_Comtype_NA_5
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_Comtype_NA_1
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_Comtype_NA_3
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_Comtype_NA_3

7 Functional specification

7.1 General issues

[SWS_Comtype_00004] ⌈It is not allowed to add any project or supplier specific extension to this file. Any extension invalidates the AUTOSAR conformity. ⌋( )

[SWS_Comtype_00015] ⌈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] ⌈The value of PduIdType and PduLengthType shall be derived from the ‘PduIdTypeEnum’ and ‘PduLengthTypeEnum’ of the EcuCPduCollection container respectively. ⌋( )

8 API specification

8.1 Type definitions

8.1.1 PduIdType

[SWS_COMTYPE_00005] [

Name:	PduIdType		
Type:	uint8, uint16		
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:	ComStackTypes.h		

] ()

[SWS_Comtype_00006][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][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][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] [

Name:	PduLengthType		
Type:	uint8, uint16, uint32		
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:	`ComStackTypes.h`		

] ()

[SWS_Comtype_00010][Variables of this type serve as length information of a PDU. The length information is provided in number of bytes.
] ()

[SWS_Comtype_00017][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] [

Name:	`PduInfoType`		
Type:	`Structure`		
Element:	`uint8*`	`SduDataPtr`	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.
	`uint8*`	`MetaDataPtr`	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.
	`PduLengthType`	`SduLength`	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:	`ComStackTypes.h`		

] ()

8.1.4 PNCHandleType

[SWS_COMTYPE_00036] [

Name:	`PNCHandleType`
Type:	`uint8`
Description:	Used to store the identifier of a partial network cluster.
Available via:	`ComStackTypes.h`

] ()

8.1.5 TPParameterType

[SWS_COMTYPE_00031] [

Name:	TPParameterType		
Type:	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:	ComStackTypes.h		

] ()

8.1.6 BufReq_ReturnType

[SWS_COMTYPE_00012] [

Name:	BufReq_ReturnType		
Type:	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:	ComStackTypes.h		

] ()

8.1.7 TpDataStateType

[SWS_COMTYPE_00027] [

Name:	TpDataStateType		
Type:	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 TxTpDataCnt 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:	`ComStackTypes.h`

] ()

8.1.8 RetryInfoType

[SWS_COMTYPE_00037] [

Name:	`RetryInfoType`		
Type:	`Structure`		
Element:	`TpDataStateType`	`TpDataState`	The enum type to be used to store the state of Tp buffer.
	`PduLengthType`	`TxTpDataCnt`	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:	`ComStackTypes.h`		

] ()

8.1.9 NetworkHandleType

[SWS_COMTYPE_00038] [

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

] ()

8.1.10 IcomConfigIdType

[SWS_COMTYPE_00039] [

Name:	`IcomConfigIdType`
Type:	`uint8`
Description:	IcomConfigIdType defines the configuration ID. An ID of 0 is the default configuration. An ID greater than 0 shall identify a configuration for Pretended Networking. There is more than 1 configuration possible.
Available via:	`ComStackTypes.h`

] ()

8.1.11 IcomSwitch_ErrorType

[SWS_COMTYPE_00040] [

Name:	IcomSwitch_ErrorType		
Type:	Enumeration		
Range:	ICOM_SWITCH_E_OK	0x00	The activation of Pretended Networking was successful.
	ICOM_SWITCH_E_FAILED	0x01	The activation of Pretended Networking was not successful.
Description:	IcomSwitch_ErrorType defines the errors which can occur when activating or deactivating Pretended Networking.		
Available via:	ComStackTypes.h		

] ()

8.2 Function definitions

Not applicable.

9 Sequence diagrams

Not applicable.

10 Configuration specification

10.1 Published parameters

For details refer to the chapter 10.3 “Published Information” in “SWS_BSWGeneral” [12].

11 Not applicable requirements

[SWS_Comtype_NA_0] [This specification item references requirements that are not applicable, because ComStack_Types neither has configurable parameters nor has reference to configuration parameters from other modules.] (SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00345, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00171, SRS_BSW_00380, SRS_BSW_00381, SRS_BSW_00412, 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_00342, SRS_BSW_00343, SRS_BSW_00160, SRS_BSW_00408, SRS_BSW_00346, SRS_BSW_00158, SRS_BSW_00401)

[SWS_Comtype_NA_1] [This specification item references requirements that are not applicable, because ComStack_Types has no interdepencies to SW Components.] (SRS_BSW_00170, SRS_BSW_00168, SRS_BSW_00423)

[SWS_Comtype_NA_2] [This specification item references requirements that are not applicable, because ComStack_Types does not implement any interrupts, is not a driver or MCAL abstraction layer or has any direct access to OS.] (SRS_BSW_00375, SRS_BSW_00101, SRS_BSW_00406, SRS_BSW_00416, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00314, SRS_BSW_00410, SRS_BSW_00314, SRS_BSW_00361, SRS_BSW_00172,)

[SWS_Comtype_NA_3] [This specification item references requirements that are not applicable, because ComStack_Types does not implement any version check information, main function. APIs, standard types,] (SRS_BSW_00323, SRS_BSW_00415, SRS_BSW_00007, SRS_BSW_00300, SRS_BSW_00307, SRS_BSW_00310, SRS_BSW_00373, SRS_BSW_00335, SRS_BSW_00411, SRS_BSW_00348, SRS_BSW_00353, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00357, SRS_BSW_00377, SRS_BSW_00304, SRS_BSW_00378, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00358, SRS_BSW_00407, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00414, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00374, SRS_BSW_00379, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334)

[SWS_Comtype_NA_4] [This specification item references requirements that are not applicable, because ComStack_Types does not have any shutdown functionality.] (SRS_BSW_00336)

[SWS_Comtype_NA_5] [This specification item references requirements that are not applicable, because ComStack_Types does not implement development errors and production errors.] (SRS_BSW_00337, SRS_BSW_00369, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00409, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00327, SRS_BSW_00350)