

<b>Document Title</b>	<b>Specification of Socket Adaptor</b>
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	416
<b>Document Classification</b>	Standard
<b>Document Status</b>	Final
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	4.3.0

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Support for decoupled data transmission</li> <li>• Optimization for Client/Server communication</li> <li>• Introduction of reliable TxConfirmations</li> <li>• Clarifications and corrections of requirements</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Clarifications and corrections of requirements</li> <li>• Editorial changes</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Introduction of IPv6 for in-vehicle communication</li> <li>• Support for Service Migration of Service Discovery Clients (SpecificRoutingGroup Handling)</li> <li>• SoAd_RequestIpAddrAssignment API extension</li> <li>• Clarifications and corrections of requirements and sequence charts</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• TP API: Harmonization of ChangeParameter function</li> <li>• Clarifications and corrections of requirements and sequence charts</li> <li>• Editorial changes</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• TP API: NotifResultType replaced by Std_ReturnType</li><li>• Clarifications and corrections of requirements and sequence charts</li><li>• Editorial changes</li><li>• Removed chapter(s) on change documentation</li></ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"><li>• APIs for SWS TCP/IP module interaction added/updated</li><li>• Functionality to trigger IPdus for sending has been added</li><li>• DoIP functionality removed</li></ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Rectify inconsistencies in synchronicity and reentrancy</li><li>• Adjust parameter multiplicity</li><li>• New traceability mechanism</li></ul>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"><li>• ComStack Harmonization.</li><li>• Allow for Post-Build Configuration</li><li>• API for IP address change notification</li><li>• Allow full handling of TCP connections</li></ul>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Initial Release Revision</li></ul>

## **Disclaimer**

This specification and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## **Advice for users**

AUTOSAR specifications may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

## Table of Contents

1	Introduction and functional overview .....	7
2	Acronyms and abbreviations .....	9
3	Related documentation.....	10
3.1	Input documents.....	10
3.2	Related standards and norms .....	11
3.3	Related specification .....	11
4	Constraints and assumptions .....	12
4.1	Limitations.....	12
4.2	Applicability to car domains.....	12
5	Dependencies on other modules.....	13
5.1	AUTOSAR TCP/IP Stack .....	13
5.2	Generic Upper Layer .....	13
5.3	File structure .....	13
5.3.1	Code file structure.....	13
5.3.2	Header file structure.....	13
5.4	Version check.....	15
6	Requirements traceability .....	16
7	Functional specification .....	20
7.1	Socket Connections .....	20
7.1.1	Socket Connection Open .....	21
7.1.2	Socket Connection Close .....	25
7.1.3	Socket Connection Open/Close Sequence Remarks.....	27
7.1.4	Notifications.....	28
7.2	PDU Transmission .....	28
7.2.1	PDU Transmission via IF-API.....	28
7.2.2	PDU Transmission via IF-API and nPduUdpTxBuffer .....	30
7.2.3	PDU Transmission via IfRoutingGroupTransmit API.....	32
7.2.4	PDU Transmission via TP-API .....	32
7.3	PDU Header option .....	34
7.4	PDU Reception.....	35
7.4.1	PDU Reception via IF-API.....	37
7.4.2	PDU Reception via TP-API (PDU Header disabled) .....	37
7.4.3	PDU Reception via TP-API (PDU Header enabled) .....	38
7.5	Best Match Algorithm .....	39
7.6	Message Acceptance Policy .....	40
7.7	TP PDU Cancelation .....	41
7.8	Disconnection and recovery .....	41
7.9	Routing Groups .....	42
7.10	PDU fan-out .....	43
7.11	Buffer handling .....	43
7.12	Error classification .....	44
7.12.1	Development Errors .....	44
7.12.2	Runtime Errors .....	44
7.12.3	Transient Faults .....	44

7.12.4	Production Errors .....	44
7.12.5	Extended Production Errors .....	44
7.13	Application notes .....	45
7.14	Debugging Concept.....	45
7.15	Version checking .....	45
8	API specification .....	46
8.1	Imported types.....	46
8.2	Type definitions .....	46
8.2.1	SoAd_SoConIdType .....	46
8.2.2	SoAd_SoConModeType .....	46
8.2.3	SoAd_RoutingGroupIdType .....	47
8.2.4	SoAd_ConfigType .....	47
8.2.5	SoAd_MeasurementIdxType.....	47
8.3	Function definitions .....	48
8.3.1	General .....	48
8.3.2	Normal Operation.....	49
8.3.3	Transmit/Receive Cancellation API.....	52
8.3.4	Information and Control API.....	53
8.4	Call-back notifications .....	72
8.4.1	SoAd_RxIndication.....	72
8.4.2	SoAd_CopyTxData .....	73
8.4.3	SoAd_TxConfirmation .....	74
8.4.4	SoAd_TcpAccepted .....	75
8.4.5	SoAd_TcpConnected .....	76
8.4.6	SoAd_TcpIpEvent .....	76
8.4.7	SoAd_LocalIpAddrAssignmentChg .....	77
8.5	Scheduled functions.....	78
8.5.1	Terms and definitions.....	78
8.5.2	SoAd_MainFunction .....	78
8.6	Expected Interfaces.....	78
8.6.1	Mandatory Interfaces .....	78
8.6.2	Optional Interfaces .....	80
8.6.3	Configurable interfaces .....	80
9	Sequence diagrams and Transition Tables .....	87
9.1	Address – Assignment .....	87
9.2	Socket Connection Setup – UDP .....	88
9.3	Socket Connection Setup – TCP .....	90
9.4	Reception – Upper Layer If API.....	92
9.5	Reception – Upper Layer TP API .....	93
9.6	Transmission – Upper Layer If API – TCP .....	95
9.7	Transmission – Upper Layer If API – UDP .....	97
9.8	Transmission – Upper Layer TP API.....	98
10	Configuration specification .....	100
10.1	How to read this chapter .....	100
10.2	Containers and configuration parameters .....	101
10.2.1	SoAd .....	101
10.2.2	SoAdBswModules .....	102
10.2.3	SoAdGeneral.....	105

10.2.4	SoAdConfig .....	109
10.2.5	SoAdSocketConnectionGroup .....	109
10.2.6	SoAdSocketConnection .....	114
10.2.7	SoAdSocketProtocol .....	115
10.2.8	SoAdSocketUdp .....	115
10.2.9	SoAdSocketTcp .....	118
10.2.10	SoAdSocketRemoteAddress .....	123
10.2.11	SoAdSocketRoute .....	124
10.2.12	SoAdSocketRouteDest .....	126
10.2.13	SoAdPduRoute .....	127
10.2.14	SoAdPduRouteDest .....	130
10.2.15	SoAdRoutingGroup .....	132
10.3	Published Information .....	134
11	Not applicable requirements .....	135

# 1 Introduction and functional overview

This document specifies the functionality, API and the configuration of the AUTOSAR Basic Software module Socket Adaptor (SoAd).

The TCP/IP concept of data transmission, particularly using Ethernet as the physical layer, has been established as a de-facto standard in the computing and telecommunication environments. The addressing of applications, logical addressing of end points and physical addressing are all covered in a layered suite of protocols and number assignments. Dynamic configuration and routing are at the core of the concepts implemented here.

AUTOSAR follows a concept of static communication relations pre-determined at compile time and rigid during run-time. The data transmitted is considered just as pre-determined as the source and sink that it needs to travel from and to.

The Socket Adaptor module aims at bridging the gap between these two concepts. By establishing a pre-determined configuration that includes the information required for AUTOSAR and leaving some items open to be updated during run-time the conflicting concepts are leveraged. Furthermore the SoAd decouples the call-back based software architecture from the socket based communication handling in the TCP/IP world.

The main purpose of the SoAd module is to create an interface between an AUTOSAR communication service module using PDUs (e.g. PDU Router) and a socket based TCP/IP stack. It will map I-PDU IDs to socket connections and vice versa. The TCP/IP protocol stack is specified in Tcplp SWS as shown in Figure 1. The internal functional structure of the TCP/IP stack is shown schematically for information purposes.

The SoAd Module, and thereby the Ethernet communication stack, was first introduced in AUTOSAR R4.0.1, some major conceptual changes have been applied between AUTOSAR R4.0.3 and AUTOSAR R4.1.1.

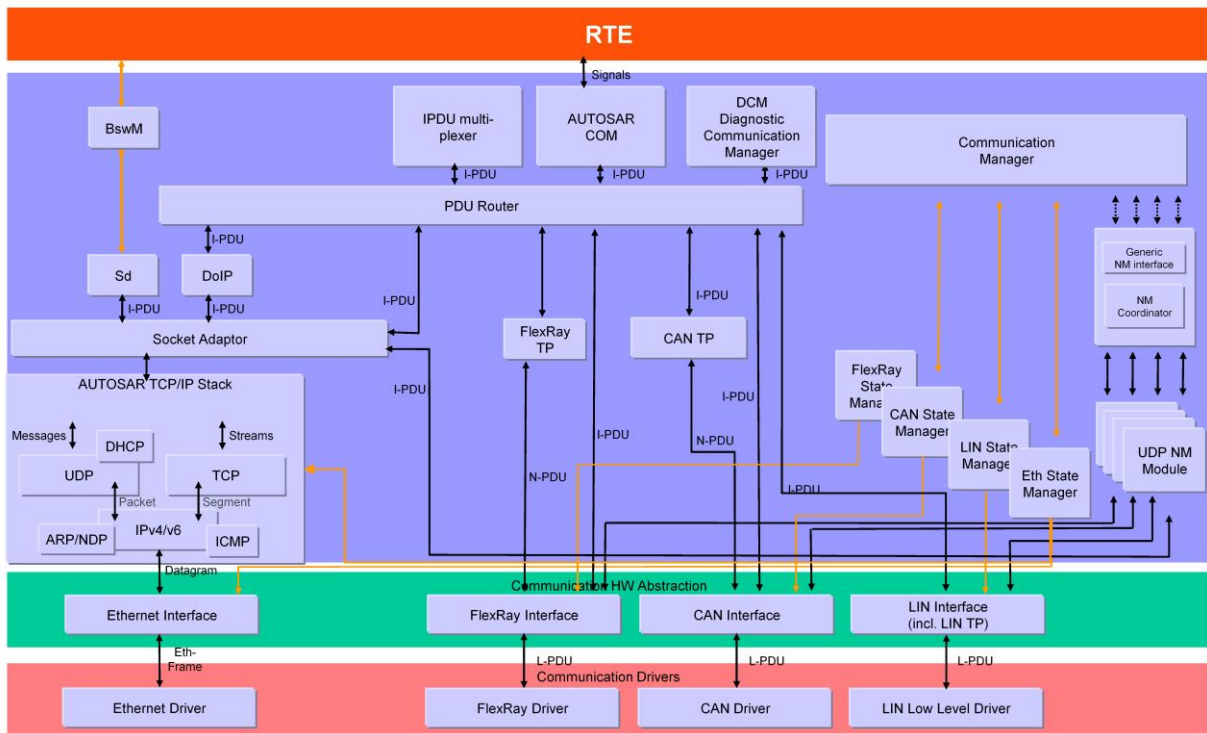


Figure 1: Extended AUTOSAR Communication Stack.



## 2 Acronyms and abbreviations

<b>Abbreviation Acronym:</b>	<b>Description:</b>
ARP	Address Resolution Protocol
DEM	Diagnostic Event Manager
DET	Default Error Tracer
DHCPv4	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol for Internet Protocol Version 6
DoIP	Diagnostics over IP
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMPv4	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol for Internet Protocol Version 6
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
NDP	Neighbor Discovery Protocol
Sd	Service Discovery
TCP	Transmission Control Protocol
TCP/IP	A family of communication protocols used in computer networks
TP	Transport Protocol
UDP	User Datagram Protocol
UdpNm	AUTOSAR UDP Network Management

<b>Term:</b>	<b>Description:</b>
<b>AUTOSAR Connector</b>	The SoAd serves as a (De)Multiplexer between different PDU sources/suppliers and the TCP/IP stack as well as between the TCP/IP stack and different PDU sinks/consumers. The term AUTOSAR connector refers to a source/supplier or sink/consumer of a PDU.
<b>TCP socket connection</b>	A socket connection which uses TCP as transport protocol (choice container SoAdSocketProtocol contains a SoAdSocketTcp subcontainer).
<b>UDP socket connection</b>	A socket connection which uses UDP as transport protocol (choice container SoAdSocketProtocol contains a SoAdSocketUdp subcontainer).
<b>IF-PDU</b>	An IF-PDU is a PDU which is sent/received via the IF-API of the SoAd
<b>TP-PDU</b>	An TP-PDU is a PDU which is send/received via the TP-API of the SoAd

## 3 Related documentation

### 3.1 Input documents

- [1] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [2] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [3] Specification of Communication Stack Types  
AUTOSAR\_SWS\_CommunicationStackTypes.pdf
- [4] Specification of ECU Configuration  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [5] Specification of BSW Scheduler  
AUTOSAR\_SWS\_BSW\_Scheduler.pdf
- [6] Specification of Default Error Tracer  
AUTOSAR\_SWS\_DefaultErrorTracer.pdf
- [7] Basic Software Module Description Template  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [8] Specification of UDP Network Management  
AUTOSAR\_SWS\_UDPNetworkManagement.pdf
- [9] Requirements on Ethernet  
AUTOSAR\_SRS\_Ethernet.pdf
- [10] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList
- [11] Specification of Service Discovery  
AUTOSAR\_SWS\_ServiceDiscovery.pdf
- [12] Specification of PDU Router  
AUTOSAR\_SWS\_PDURouter.pdf
- [13] Specification of TCP/IP Stack  
AUTOSAR\_SWS\_TCPIP.pdf
- [14] Specification of Module XCP  
AUTOSAR\_SWS\_XCP.pdf
- [15] Specification of Diagnostics over IP  
AUTOSAR\_SWS\_DoIP.pdf

- [16] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### **3.2 Related standards and norms**

- [1] IETF RFC 4702  
<http://tools.ietf.org/html/rfc4702>

- [2] IETF RFC 4704  
<http://tools.ietf.org/html/rfc4704>

### **3.3 Related specification**

AUTOSAR provides a General Specification on Basic Software modules [16] (SWS BSW General), which is also valid for Socket Adaptor.

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

## 4 Constraints and assumptions

### 4.1 Limitations

The transmission of data using TCP/IP over Ethernet requires about 60 bytes of header information. This implies that for small messages the header overhead may reach an unacceptably high percentage.

To avoid further protocol overhead, the use of a single socket connection per PDU is described here. However, this solution is very resource intensive, particularly if many small PDUs are to be transmitted. One solution described here as an option is to add a small PDU header, containing an ID and length information. This enables transmission of multiple PDUs via one socket connection. Additionally a resource conservation scheme is included in this specification as an option.

This document does not cover the assignment of UDP or TCP port numbers. There is no reserved space within the IANA assigned number range. Each implementer is responsible for managing the used port numbers.

This document does not cover the management of IP addresses. This might be done dynamically, e.g. by using DHCP, or statically. It is the implementers responsibility to prevent address conflicts and achieve compliance with IANA address assignments.

This specification does not prescribe a certain physical layer or data rate.

### 4.2 Applicability to car domains

No restrictions.

## 5 Dependencies on other modules

This section outlines relations between the SoAd and related AUTOSAR basic software modules. It contains brief descriptions of the services required by the SoAd from other modules and how other modules will call the SoAd.

### 5.1 AUTOSAR TCP/IP Stack

The Tcplp module implements the main protocols of the TCP/IP protocol family (TCP, UDP, IPv4, ARP, ICMP, DHCP, IPv6, NDP, ICMPv6, DHCPv6) and provides dynamic, socket based communication via Ethernet. The SoAd module is one of the possible upper layer modules of the Tcplp module.

### 5.2 Generic Upper Layer

The SoAd module provides a generic upper layer support, i.e. the SoAd offers its services to any upper layer which conforms to the SoAd generic upper layer API/configuration. Each upper layer of the SoAd may specify which kind of service it wants to use.

In the AUTOSAR architecture a number of SoAd upper layer modules are already defined. The following list specifies these module and provides a rough description of the SoAd services used:

- PDU Router (PduR) [12]: IF-PDU and TP-PDU API
- UDP Network Management (UdpNm) [8]: IF-PDU API
- XCP on Ethernet (Xcp) [14] : IF-PDU API
- Service Discovery (Sd) [11]: IF-PDU API, Control API (8.3.4)
- Diagnostics over IP (DoIP) [15]: IF-PDU and TP-PDU API, Control API (8.3.4)

### 5.3 File structure

#### 5.3.1 Code file structure

For details refer to the chapter 5.1.6 “Code file structure” in *SWS\_BSWGeneral*.

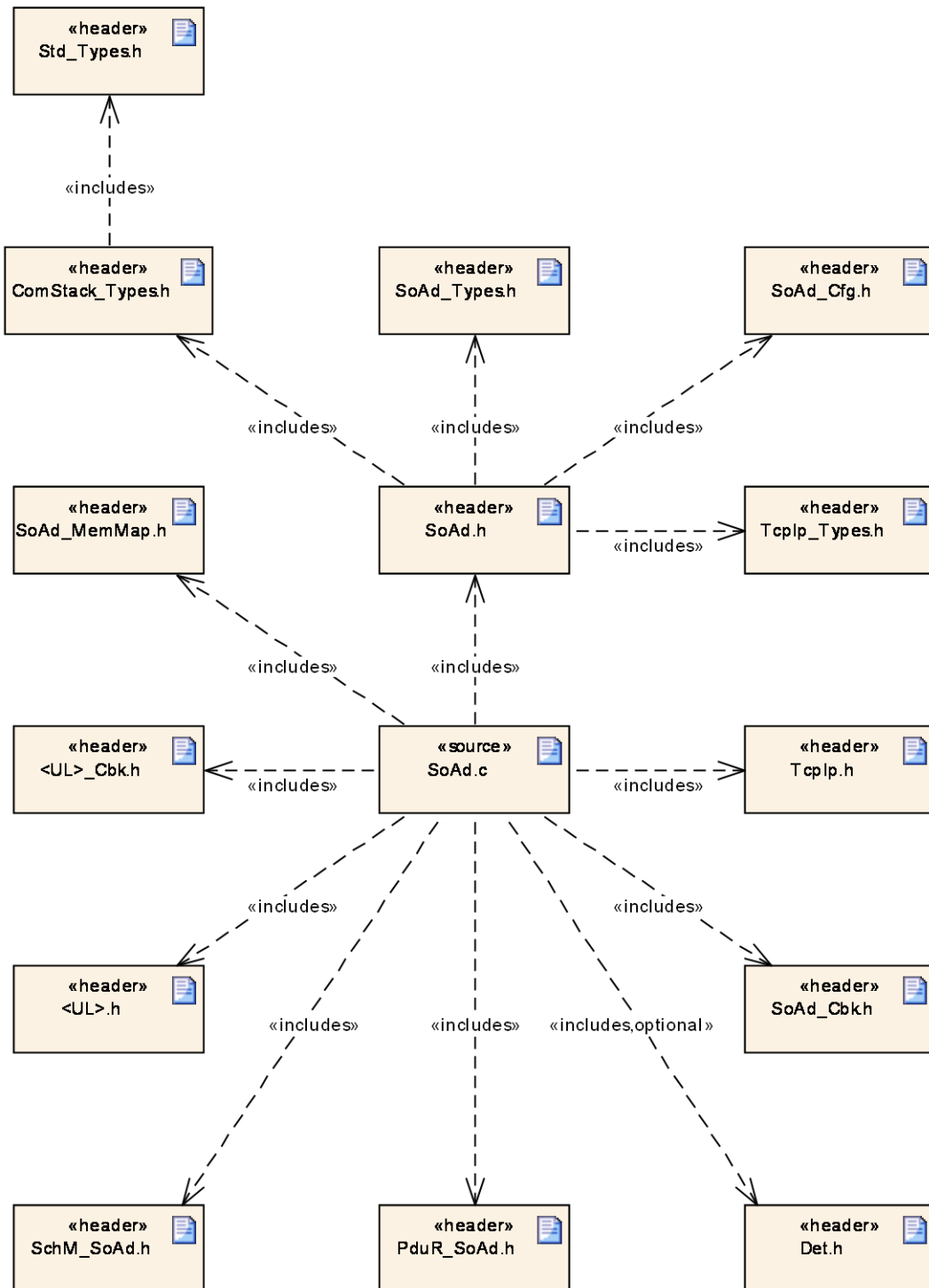
#### 5.3.2 Header file structure

**[SWS\_SoAd\_00072]** [The SoAd module shall provide the following H-files:

- `SoAd.h` (for declaration of provided interface functions)
- `SoAd_Types.h` (for public types defined by SoAd) | ( )

**[SWS\_SoAd\_00073]** [The SoAd module shall include the following H-files of other modules:

- TcpIp.h – header file of the AUTOSAR TCP/IP stack
- ComStack\_Types.h [3]
- <UL>\_Cb.h (for callback functions of the SoAd upper layer modules) ] ( )



Figure

2: AUTOSAR SoAd header file structure.

## 5.4 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

## 6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00005	Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_SoAd_00296
SRS_BSW_00006	The source code of software modules above the $\mu$ C Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_SoAd_00296
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_SoAd_00296
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_SoAd_00296
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_SoAd_00296
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_SoAd_00296
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_SoAd_00296
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_SoAd_00296
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_SoAd_00296
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_SoAd_00296
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler	SWS_SoAd_00296



	and platform independent	
SRS_BSW_00307	Global variables naming convention	SWS_SoAd_00296
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_SoAd_00296
SRS_BSW_00312	Shared code shall be reentrant	SWS_SoAd_00296
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_SoAd_00296
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_SoAd_00296
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_SoAd_00296
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_SoAd_00296
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_SoAd_00296
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_SoAd_00296
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_SoAd_00296
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_SoAd_00296
SRS_BSW_00335	Status values naming convention	SWS_SoAd_00296
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_SoAd_00296
SRS_BSW_00337	Classification of development errors	SWS_SoAd_00732, SWS_SoAd_00744
SRS_BSW_00341	Module documentation shall contain all needed informations	SWS_SoAd_00296

SRS_BSW_00347	A Naming separation of different instances of BSW drivers shall be in place	SWS_SoAd_00296
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_SoAd_00296
SRS_BSW_00410	Compiler switches shall have defined values	SWS_SoAd_00296
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_SoAd_00296
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_SoAd_00296
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_SoAd_00296
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_SoAd_00296
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_SoAd_00296
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_SoAd_00296
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_SoAd_00296
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_SoAd_00296
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_SoAd_00296
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_SoAd_00296
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_SoAd_00296
SRS_Eth_00058	SoAd shall support generic	SWS_SoAd_00741

	upper layers		
SRS_Eth_00085	Robustness against the change of logical addresses	SWS_SoAd_00532, SWS_SoAd_00695, SWS_SoAd_00743, SWS_SoAd_00746	SWS_SoAd_00694, SWS_SoAd_00742, SWS_SoAd_00745,
SRS_Eth_00116	The Socket Adaptor shall implement a mechanism to transmit multiple PDUs within the same UDP datagram	SWS_SoAd_00546, SWS_SoAd_00548, SWS_SoAd_00550, SWS_SoAd_00684, SWS_SoAd_00690, SWS_SoAd_00696, SWS_SoAd_00734, SWS_SoAd_00736,	SWS_SoAd_00547, SWS_SoAd_00549, SWS_SoAd_00683, SWS_SoAd_00685, SWS_SoAd_00691, SWS_SoAd_00697, SWS_SoAd_00735, SWS_SoAd_00747
SRS_Eth_00124	The SoAd shall implement mechanisms to share the same PDU pair for the reception from and transmission to multiple remote nodes	SWS_SoAd_00738, SWS_SoAd_00740	SWS_SoAd_00739,
SRS_Eth_00131	The SoAd shall support access to measurement counter values	SWS_SoAd_00748, SWS_SoAd_00750, SWS_SoAd_00752, SWS_SoAd_00754, SWS_SoAd_00756, SWS_SoAd_00758	SWS_SoAd_00749, SWS_SoAd_00751, SWS_SoAd_00753, SWS_SoAd_00755, SWS_SoAd_00757,

## 7 Functional specification

The SoAd enables PDU-based communication via the TCP/IP network. Therefore AUTOSAR I-PDUs are mapped to socket connections which are configured and maintained by SoAd. To use a socket connection for more than one I-PDU a SoAd PDU Header can optionally be added in front of each I-PDU. A message acceptance policy is specified to define which TCP connections and UDP datagrams from remote nodes are accepted. Socket connections can be opened automatically or manually via a request from the upper layer. For disconnection and recovery of a socket connection a policy is defined. An upper layer of the SoAd can use the IF-API as well as the TP-API for transmission and reception of PDUs. To selectively enable/disable the routing of PDUs from or to socket connections PDU routing groups are defined and can be controlled by the upper layer of the SoAd. An IF-PDU can also be forwarded to multiple socket connections or a message received from a socket connection can be forwarded as different IF-PDUs to the same or different upper layers of the SoAd (PDU Fan-out).

Note: The SoAd Module does not provide any means for adaption of the Bit- or Byte-Order (Endianness) within a PDU.

### 7.1 Socket Connections

The TCP/IP communication is based on internet sockets. An internet socket is the endpoint of a communication link which is identified by the tuple IP address and port. Depending on the transport protocol sockets are separated in UDP sockets for connection-less communication via the User Datagram Protocol (UDP) and TCP sockets for connection-oriented communication via the Transmission Control Protocol (TCP). TCP is based on point-to-point communication relations. A broadcast or multicast is not possible in TCP. TCP requires for one party to establish the connection and for the other to accept the incoming request. Two stations may establish multiple connections with each other, each will be handled by a different socket and need to differ at least in one of the port numbers used. In TCP all messages sent from a source to a sink are considered a continuous stream of consecutive bytes with preserved order. An acknowledgement scheme is in place to preserve the byte order spanning all messages. Messages are retransmitted by the source if the sink does not acknowledge reception within a certain time. TCP ensures data integrity (using checksums), byte order and completeness.

For the abstraction of the TCP/IP communication SoAd defines socket connections. A SoAd socket connection specifies a connection between a local socket (i.e. local address identifier and local port) and a remote socket (i.e. remote IP address and port), as well as connection parameters like transport protocol, usage of the SoAd PDU Header, buffer requirements, connection setup, transport protocol related parameters and so on. Each socket connection can be identified by a unique identifier (SoConId). For the simultaneous support of multiple communication partners per local socket, socket connections with identical connection parameters can be grouped to socket connection groups.

[SWS\_SoAd\_00588] SoAd shall store a request to open or close a socket connection when called with `SoAd_OpenSoCon()` and `SoAd_CloseSoCon()` respectively, but handle the request only in the `SoAd_MainFunction()` respecting the connection setup and shutdown policy.] ()

[SWS\_SoAd\_00743] SoAd shall lock the remote address during the following situations:

- (a) TCP socket connections not in state `SOAD_SOCON_OFFLINE`,
- (b) active receptions,
- (c) pending receptions of TP-PDUs,
- (d) active transmissions,
- (e) pending transmissions of the nPdu feature,
- (f) pending transmissions initiated via `SoAd_IfRoutingGroupTransmit()` or `SoAd_IfSpecificRoutingGroupTransmit()`.

A locked remote address can't be modified by upper layers.] (SRS\_Eth\_00085)

### 7.1.1 Socket Connection Open

[SWS\_SoAd\_00589] In the `SoAd_MainFunction()`, SoAd shall try to open each socket connection which fulfills all of the following criterias:

- (1) No `Tcplp` socket is assigned to the socket connection
- (2) Open is either (a) explicitly requested by a previous `SoAd_OpenSoCon()` call which has not been revoked by a following `SoAd_CloseSoCon()` call or (b) implicitly requested when `SoAdSocketAutomaticSoConSetup` is `TRUE`
- (3) remote address is set (either specified by configuration or set via the function `SoAd_SetRemoteAddr()`)
- (4) local IP address is assigned, i.e. `SoAd_LocalIpAddressAssignmentChg()` has been called with the related `LocalAddrId` and `TCPIP_IPADDR_STATE_ASSIGNED` as State.

] ()

[SWS\_SoAd\_00590] SoAd shall perform the following actions within `SoAd_MainFunction()` to open either a UDP socket connection which is part of a socket connection group containing a single socket connection (i.e. there is only one socket connection in the socket connection group configuration container) or a TCP socket connection with `SoAdSocketTcplnInitiate` set to `TRUE`:

- (1) Get an appropriate socket from `Tcplp` by calling `TcpIp_SoAdGetSocket()` with the `Tcplp_DomainType` implicitly specified by `SoAdSocketLocalAddressRef`, and the protocol type specified by `SoAdSocketProtocol`.
- (2) Change the socket specific parameters according to [SWS\_SoAd\_00689]
- (3) Bind the socket to the local address and port by calling `TcpIp_Bind()` with the local address identifier specified by `SoAdSocketLocalAddressRef` and local port specified by `SoAdSocketLocalPort`.
- (4) In case of a TCP socket initiate the TCP connection by calling `Tcplp_TcpConnect()`.

] ()

[SWS\_SoAd\_00638] SoAd shall perform the following actions within `SoAd_MainFunction()` to open a TCP socket connection with `SoAdSocketTcpInitiate` set to `FALSE`:

- (1) In case no Listen-Socket is assigned to the socket connection:
  - (a) Get an appropriate socket from `Tcplp` by calling `TcpIp_SoAdGetSocket()` with the `Tcplp_DomainType` implicitly specified by `SoAdSocketLocalAddressRef`, and the protocol type specified by `SoAdSocketProtocol`.
  - (b) Change the socket specific parameters according to [SWS\_SoAd\_00689]
  - (c) Bind the socket to the local address and port by calling `TcpIp_Bind()` with the local address identifier specified by `SoAdSocketLocalAddressRef` and local port specified by `SoAdSocketLocalPort`.
  - (d) Assign the Listen-Socket to the socket connection group
  - (e) Activate the socket connection to accept connections from remote nodes
  - (f) Listen for a remote connection requests on the Listen-Socket by calling `Tcplp_TcpListen()` with `MaxChannels` set to the number of socket connections that are part of the TCP socket connection group
- (2) In case the Listen-Socket is already assigned to the socket connection:
  - (a) Activate the socket connection to accept connections from remote nodes

] ()

Note: all socket connections of a TCP socket connection group (and `SoAdSocketTcpInitiate` set to `FALSE`) share one `Tcplp` socket for incoming connection requests ("Listen-Socket"), but use a separate `Tcplp` socket created by the `Tcplp` module and provided via `SoAd_TcpAccepted()` after the connection has been establishment.

[SWS\_SoAd\_00639] SoAd shall perform the following actions within `SoAd_MainFunction()` to open a UDP socket connection which is part of a socket connection group containing multiple socket connections (i.e. there is more than one socket connection in the socket connection group configuration container):

- (1) In case no UDP socket is assigned to the socket connection group:
  - (a) Get an appropriate socket from `Tcplp` by calling `TcpIp_SoAdGetSocket()` with the domain type implicitly specified by `SoAdSocketLocalAddressRef`, and the protocol type specified by `SoAdSocketProtocol`.
  - (b) Change the socket specific parameters according to [SWS\_SoAd\_00689]
  - (c) Bind the socket to the local address and port by calling `TcpIp_Bind()` with the local address identifier specified by `SoAdSocketLocalAddressRef` and local port specified by `SoAdSocketLocalPort`.
  - (d) Assign the UDP socket to the socket connection group
  - (e) Activate the socket connection for communication via the shared UDP socket of the socket connection group
- (2) In case the UDP socket is already assigned to the socket connection group:
  - (a) Activate the socket connection for communication via the shared UDP socket of the socket connection group

] ()

Note: all socket connections of a UDP socket connection group share the same `Tcplp` socket.



[SWS\_SoAd\_00689] In case socket related parameters shall be changed as part of allocating a new socket, SoAd shall change the parameters according to the configuration of the associated socket connection by calling `TcpIp_ChangeParameter()` with `ParameterId` and `ParameterValue` for each related clause as specified below:

- (1) In case of a TCP socket: `TCPIP_PARAMID_TCP_RXWND_MAX` and the value specified by `SoAdSocketTpRxBufferMin` if the optional parameter is enabled
- (2) `TCPIP_PARAMID_FRAMEPRIO` and the value specified by `SoAdSocketFramePriority` if the optional parameter is enabled
- (3) In case of a TCP socket: `TCPIP_PARAMID_TCP_NAGLE` and the value `0x01` if the related optional parameter `SoAdSocketTcpNoDelay` is set to `FALSE` or `0x00` if the parameter is set to `TRUE`.
- (4) In case of a TCP socket: `TCPIP_PARAMID_TCP_KEEPALIVE` and the value specified by `SoAdSocketTcpKeepAlive`
- (5) In case of a TCP socket: `TCPIP_PARAMID_TCP_KEEPALIVE_TIME` and the value specified by `SoAdSocketTcpKeepAliveTime` if the optional parameter is enabled
- (6) In case of a TCP socket: `TCPIP_PARAMID_TCP_KEEPALIVE_PROBES_MAX` and the value specified by `SoAdSocketTcpKeepAliveProbesMax` if the optional parameter is enabled
- (7) In case of a TCP socket: `TCPIP_PARAMID_TCP_KEEPALIVE_INTERVAL` and the value specified by `SoAdSocketTcpKeepAliveInterval` if the optional parameter is enabled.]
- (8) In case of a TCP socket: `TCPIP_PARAMID_TCP_OPTIONFILTER` and the value of `TcpIpTcpConfigOptionFilterId` specified in `TcpIpTcpConfigOptionFilter` referenced by `SoAdSocketTCPOptionFilterRef` if the optional parameter is enabled.
- (9) `TCPIP_PARAMID_PATHMTU_ENABLE` and the value specified by `SoAdSocketPathMTUEnable` if the optional parameter is enabled.
- (10) `TCPIP_PARAMID_FLOWLABEL` and the value specified by `SoAdSocketFlowLabel` if the optional parameter is enabled.
- (11) `TCPIP_PARAMID_DSCP` and the value specified by `SoAdSocketDifferentiatedServicesField` if the optional parameter is enabled.
- (12) In case of a UDP socket: `TCPIP_PARAMID_UDP_CHECKSUM` and the value of `SoAdSocketUDPChecksumEnabled`. ()

[SWS\_SoAd\_00591] Within `SoAd_MainFunction()` and after successfully performing the open actions, SoAd shall change the state of the socket connection to `SOAD_SOCON_ONLINE` in case of a UDP socket and either `SoAdSocketUdpListenOnly` is set to `TRUE` or a remote address is set by a value that does not contain any wildcards.] ()

[SWS\_SoAd\_00686] Within `SoAd_MainFunction()` and after successfully performing the open actions, SoAd shall change the state of the socket connection to `SOAD_SOCON_RECONNECT` in case of

- (1) a TCP socket connection or

- (2) a UDP socket connection that is configured with a remote address containing wildcards.] ()

[SWS\_SoAd\_00592] Within SoAd\_RxIndication() and before analyzing or forwarding of any message data, SoAd shall (a) overwrite the remote address parts specified with wildcards (e.g. remote IP address set to TCPIP\_IPADDR\_ANY) with the related source address parts of the received message and (b) change the state of the socket connection to SOAD\_SOCON\_ONLINE in case all of the following conditions are true:

- (1) Current connection state is not SOAD\_SOCON\_ONLINE
- (2) UDP socket
- (3) SoAdSocketUdpListenOnly is set to FALSE
- (4) SoAdSocketMsgAcceptanceFilterEnabled is set to TRUE
- (5) Remote address is set, but contains wildcards
- (6) Received message is accepted according to the message acceptance policy

] ()

[SWS\_SoAd\_00593] Within SoAd\_TcpConnected() SoAd shall change the state of the socket connection to SOAD\_SOCON\_ONLINE in case all of the following conditions are true:

- (1) Current connection state is not SOAD\_SOCON\_ONLINE
- (2) TCP socket
- (3) SoAdSocketTcpInitiate is set to TRUE

] ()

[SWS\_SoAd\_00594] At SoAd\_TcpAccepted(), SoAd shall perform the following actions if the TCP SoAdSocketConnectionGroup related to SocketId has SoAdSocketTcpInitiate set to FALSE:

- (1) choose one of the socket connections using the best match algorithm (see [SWS\_SoAd\_00680]), and either proceed with the selected socket connection or skip further processing and return with E\_NOT\_OK if no match can be found
- (2) overwrite the remote address parts specified with wildcards (e.g. remote IP address set to TCPIP\_IPADDR\_ANY) with the related source address parts of the received message if the remote address set for the socket connection contains wildcards
- (3) assign the TcpIp socket used for the established connection and provided as parameter SocketIdConnected to the chosen socket connection,
- (4) change the state of this socket connection to SOAD\_SOCON\_ONLINE and return E\_OK

] ()

[SWS\_SoAd\_00636] At SoAd\_TcpAccepted(), SoAd shall perform the following actions if the TCP SoAdSocketConnectionGroup related to SocketId has both SoAdSocketTcpInitiate and SoAdSocketMsgAcceptanceFilterEnabled set to FALSE and is not online (i.e. current connection state not SOAD\_SOCON\_ONLINE):

- (1) assign the TcpIp socket used for the established connection and provided as parameter SocketIdConnected to the socket connection and



- (2) change the state of the socket connection to SOAD\_SOCON\_ONLINE and return E\_OK.

] ()

[SWS\_SoAd\_00595] For socket connection with PDU Header mode disabled (SoAdPduHeaderEnable = FALSE) and an upper layer with TP-API, SoAd shall call <Up>\_[SoAd][Tp]StartOfReception() with TpSduLength = 0 at the end of the connection setup.] ()

### 7.1.2 Socket Connection Close

[SWS\_SoAd\_00604] In the SoAd\_MainFunction(), SoAd shall close each socket connection which fulfills all of the following criteria:

- (1) Current connection state is not SOAD\_SOCON\_OFFLINE
- (2) Close is explicitly requested by a previous SoAd\_CloseSoCon() call
- (3) No upper layer requested to keep the socket connection open at the time of the SoAd\_CloseSoCon() call (i.e. SoAd\_CloseSoCon() has been called as often as SoAd\_OpenSoCon()) or SoAd\_CloseSoCon() has been called with abort set to TRUE.

] ()

[SWS\_SoAd\_00637] SoAd shall perform the following actions within SoAd\_MainFunction() to close a socket connection:

- (1) Terminate active TP sessions (if any) and notify the upper layer about the termination
- (2) Disable further transmission or reception for this socket connection, i.e. new transmit requests shall be rejected with E\_NOT\_OK and received messages shall simply be discarded.
- (3) Close related TcpIp sockets
- (4) Change the state of the socket connection to SOAD\_SOCON\_OFFLINE if closing of the socket connection results from a SoAd\_CloseSoCon() request or to SOAD\_SOCON\_RECONNECT otherwise.

] ()

[SWS\_SoAd\_00640] To notify the upper layer about the termination of an active TP transmission on closing a socket connection within SoAd\_MainFunction(), SoAd shall call <Up>\_[SoAd][Tp]TxConfirmation() with parameter result set to

- (1) E\_OK if disconnect is caused by SoAd\_CloseSoCon() and all data was correctly transmitted, and
- (2) E\_NOT\_OK for any other cause.

] ()

[SWS\_SoAd\_00641] To notify the upper layer about the termination of an active TP reception on closing a socket connection within SoAd\_MainFunction(), SoAd shall call <Up>\_[SoAd][Tp]RxIndication() with parameter result set to

- (1) E\_OK if disconnection is caused by SoAd\_CloseSoCon() and all received data was correctly delivered to the upper layer, and

(2) E\_NOT\_OK for any other cause.

] ()

[SWS\_SoAd\_00642] To close related TcpIp sockets on closing a socket connection within SoAd\_MainFunction(), SoAd shall perform the following actions:

(1) In case of a TCP socket connection:

- (a) Close the related socket by calling `TcpIp_CloseSocket()` with parameter `abort` set to the same value as provided by `SoAd_CloseSoCon()` or set to `FALSE` in case closing was not initiated by `SoAd_CloseSoCon()`.
- (b) If all socket connections of a TCP socket connection group have been closed by `SoAd_CloseSoCon()`: Close the related Listen-Socket by calling `TcpIp_CloseSocket()` with parameter `abort` set to the same value as provided by `SoAd_CloseSoCon()` or set to `FALSE` in case closing was not initiated by `SoAd_CloseSoCon()`.

(2) In case of a UDP socket connection:

- (a) If the socket connection is NOT part of a socket connection group (i.e. there is only one socket connection in the socket connection group configuration container): Close the related socket by calling `TcpIp_CloseSocket()` with parameter `abort` set to the same value as provided by `SoAd_CloseSoCon()` or set to `FALSE` in case closing was not initiated by `SoAd_CloseSoCon()`.
- (b) If all socket connections of a UDP socket connection group have been closed by `SoAd_CloseSoCon()`: Close the related UDP socket by calling `TcpIp_CloseSocket()` with parameter `abort` set to the same value as provided by `SoAd_CloseSoCon()` or set to `FALSE` in case closing was not initiated by `SoAd_CloseSoCon()`.

] ()

[SWS\_SoAd\_00643] Within `SoAd_TcpIpEvent()` with `Event` set to `TCPIP_UDP_CLOSED`, SoAd shall

- (1) remove the assignment of the TcpIp socket identified by `SocketId` from the related UDP socket connection group and
- (2) close all socket connections of the related socket connection group that are in `SOAD_SOCON_ONLINE` (i.e. perform the specified closing actions with the exception of closing related TcpIp sockets)] ()

[SWS\_SoAd\_00645] Within `SoAd_TcpIpEvent()` with `Event` set to `TCPIP_TCP_CLOSED` for a Listen-Socket, SoAd shall remove the assignment of the TcpIp socket identified by `SocketId` from the related TCP socket connection group.] ()

[SWS\_SoAd\_00646] Within `SoAd_TcpIpEvent()` with `Event` set to `TCPIP_TCP_CLOSED` or `TCPIP_TCP_RESET`, SoAd shall

- (1) remove the assignment of the TcpIp socket identified by `SocketId` from the related socket connection and
- (2) close the socket connection if it is in `SOAD_SOCON_ONLINE` (i.e. perform the specified closing actions with the exception of closing related TcpIp socket). ] ()

[SWS\_SoAd\_00688] Within SoAd\_TcplpEvent() with Event set to TCPIP\_TCP\_FIN\_RECEIVED SoAd shall close the related socket by calling Tcplp\_Close() with parameter abort set FALSE.] ()

### 7.1.3 Socket Connection Open/Close Sequence Remarks

The Requirements describe in Chapters 7.1.1 and 7.1.2 shall lead to the following intended behavior:

Scenario 1:

1: Open  
2: Main - ONLINE  
3: Close  
4: Open  
5: Main - OFFLINE  
6: Main - ONLINE

Comment: Open request (4) will be executed after close request (3) has been executed.

Rational: To clearly separate two communication sessions, close has to win against open, i.e. open request (4) shall not revoke the close request (3)

Scenario 2:

1: Open  
2: Main - ONLINE  
3: Close  
4: Open  
5: Close  
6: Open  
7: Close  
8: Main - OFFLINE  
9: Main, no change

Comment: Close request (5) revokes open request (4) and (7) revokes (6)

Rational: there is no need for the communication session as the upper layer revoked it before it was ever active

### 7.1.4 Notifications

[SWS\_SoAd\_00597] Each time a socket connection state changes, SoAd shall notify the upper layer of a socket connection state change with the configured upper layer notification function `<Up>_SoConModeChg()` and the new state if `SoAdSocketSoConModeChgNotification` is set to `TRUE` for the socket connection.] ()

[SWS\_SoAd\_00741] Each time a socket connection state changes, SoAd shall notify the upper layer specified by `SoAdSocketSoConModeChgNotifUpperLayerRef` with the configured upper layer notification function `<Up>_SoConModeChg()` and the new state if the optional reference is set for the socket connection.] (SRS\_Eth\_00058)

[SWS\_SoAd\_00598] Each time the IP address assignment related to a socket connection changes, SoAd shall notify the upper layer of the IP address assignment change with the configured upper layer notification function `<Up>_LocalIpAddrAssignmentChg()` and the new address state if `SoAdSocketIpAddrAssignmentChgNotification` is set to `TRUE` for the socket connection.] ()

## 7.2 PDU Transmission

For the transmission of an upper layer module PDU via an UDP or TCP socket, the SoAd configuration specifies a PDU route which is linked to a socket connection. A PDU route (`SoAdPduRoute`, `SoAdPduRouteDest`) describes the route from an upper layer module of the SoAd to the related socket of the TcpIp stack which is described by the socket connection (`SoAdSocketConnection`, `SoAdSocketConnectionGroup`). The upper layer module of the SoAd may use the Interface (IF) API or the Transport Protocol (TP) API for the transmit request and data provision respectively.

### 7.2.1 PDU Transmission via IF-API

[SWS\_SoAd\_00539] For the transmission of a PDU requested by an upper layer using the IF-API, the SoAd shall

- (1) Identify the related socket connection and PDU route by using the `TxPduId` provided at `SoAd_IfTransmit()`.
- (2) Call the related `TcpIp` transmit function depending on the connection type if the PDU length `> 0` or `SoAdPduHeaderEnable` is `TRUE`, otherwise SoAd shall Skip further processing and return with `E_NOT_OK`.

] ()

[SWS\_SoAd\_00738] If development error detection is enabled: In case of a transmit request for a `SoAdPduRoute` that refers to a global PDU structure configured with a `MetaDatum` of the type `SOCKET_CONNECTION_ID_16` and the contained `SoAdPduRouteDest` refers to a socket connection group, SoAd shall raise the development error `SOAD_E_INV_METADATA`, if the socket connection identified

by `PduInfoType.MetadataPtr` is not part of the socket connection group of the related `SoAdPduRouteDest`.] (SRS\_Eth\_00124)

[SWS\_SoAd\_00739] In case of a transmit request for a `SoAdPduRoute` that refers to a global PDU structure configured with a `MetaDatum` of the type `SOCKET_CONNECTION_ID_16` and the contained `SoAdPduRouteDest` refers to a socket connection group, SoAd shall only perform the transmission on the socket connection identified by `PduInfoType.MetadataPtr` instead of the whole group.] (SRS\_Eth\_00124)

[SWS\_SoAd\_00540] In case of a UDP socket connection the SoAd shall (if not specified otherwise) call `TcpIp_UdpTransmit()` with `SocketId` and remote address specified in the `SocketConnection` and the PDU length specified in the `SoAd_IfTransmit()` call as `TotalLength`.] ()

[SWS\_SoAd\_00542] In case of a TCP socket connection the SoAd shall call `TcpIp_TcpTransmit()` with `SocketId` specified in the `SocketConnection`, the PDU length specified in the `SoAd_IfTransmit()` call, as `AvailableLength` and `ForceRetrieve` set to `TRUE`.] ()

Note: `TxDulId` identifies a `SoAdPduRoute` in the SoAd configuration which contains one or more `SoAdPduRouteDest` container which references to a `SoAdSocketConnection`

[SWS\_SoAd\_00543] The `TcpIp` module will retrieve the PDU data within the context of the `TcpIp` transmit call by using `SoAd_CopyTxData()` where the SoAd shall copy (the requested part of) the PDU to the memory specified by parameter `BufPtr`.] ()

[SWS\_SoAd\_00731] If `SoAd_IfTransmit` was called with `PduInfoPtr->SduDataPtr` set to `NULL_PTR`, SoAd shall use `<Up>_[SoAd][If]TriggerTransmit()` to retrieve the PDU data from the upper layer.] ()

[SWS\_SoAd\_00544] In case of a UDP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][If]TxConfirmation()`) with result set to `E_OK` within the next `SoAd_MainFunction()` after the latest `TcpIp_UdpTransmit()` call returning successfully.] ()

[SWS\_SoAd\_00545] In case of a TCP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][If]TxConfirmation()`) with result set to `E_OK` within the `SoAd_TxConfirmation()` callback function after all PDU data (from one or multiple transmit requests) have been confirmed for transmission.] ()

Note: there is only a single confirmation even in case of multiple transmit requests for the same PDU, i.e. in case a further transmit is requested for the same PDU on a

TCP socket connection before the last request is completed, there is no separate confirmation for the last request, but only a final confirmation for all PDU data.

### 7.2.2 PDU Transmission via IF-API and nPduUdpTxBuffer

[SWS\_SoAd\_00546] In case SoAdTxUdpTriggerMode is set to TRIGGER\_NEVER for any PDU route (SoAdPduRouteDest) related to a socket connection and all upper layers belonging to the related socket connection have SoAdTxUpperLayerType set to "IF", SoAd shall use the nPdu feature for this socket connection.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00547] In case the nPdu feature is used for a socket connection and TriggerMode is set to TRIGGER\_NEVER for the actual PDU (SoAdPduRouteDest), SoAd shall store the PDU for the socket connection (instead of calling `TcpIp_UdpTransmit()`).] (SRS\_Eth\_00116)

[SWS\_SoAd\_00747] In case the nPdu feature is used for a socket connection and the related PDU parameter SoAdTxPduCollectionSemantics is set to SOAD\_COLLECT\_LAST\_IS\_BEST, SoAd shall only store the transmission request instead of the PDU data. When SoAd needs to provide the PDU data, SoAd shall retrieve the data from the upper layer by calling `<Up>_[SoAd][If]TriggerTransmit()`.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00734] In case the nPdu feature is used for a socket connection, the related PDU parameter SoAdTxPduCollectionSemantics is set to SOAD\_COLLECT\_LAST\_IS\_BEST and the upper layer doesn't provide all the requested data via `<Up>_[SoAd][If]TriggerTransmit()` in the context of SoAd\_CopyTxData, SoAd shall abort the transmission and return E\_NOT\_OK.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00548] In case the nPdu feature is used for a socket connection and TriggerMode is set to TRIGGER\_ALWAYS for the current PDU (SoAdPduRouteDest) and the resulting PDU data and headers don't exceed SoAdSocketnPduUdpTxBufferMin, SoAd shall transmit all PDUs stored for the socket connection (if any) and the current PDU by calling `TcpIp_UdpTransmit()`.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00685] In case the nPdu feature is used for a socket connection and TriggerMode is set to TRIGGER\_ALWAYS for the current PDU (SoAdPduRouteDest) and the resulting PDU data and headers would exceed SoAdSocketnPduUdpTxBufferMin, SoAd shall first transmit all PDUs stored for the socket connection (if any) by calling `TcpIp_UdpTransmit()` and then the current PDU by calling `TcpIp_UdpTransmit()` once more.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00549] In case the nPdu feature is used for a socket connection and TriggerMode is set to TRIGGER\_NEVER for the current PDU (SoAdPduRouteDest) and the resulting PDU data and headers would exceed SoAdSocketnPduUdpTxBufferMin, SoAd shall first transmit all PDUs stored for the



socket connection by calling `TcpIp_UdpTransmit()` and then store the PDU for the socket connection.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00690] SoAd shall preserve the order of PDUs transmitted via a socket connection that uses the nPdu feature. Pdus collected on the sender side first shall be extracted and indicated to the receivers on the receiving side first as well.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00691] In case the nPdu feature is used for a socket connection and all PDUs are configured with `SoAdTxPduCollectionSemantics` set to `SOAD_COLLECT_QUEUED`, SoAd shall transmit all PDUs individually, also PDUs with the same PduId.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00735] In case the nPdu feature is used for a socket connection and all PDUs are configured with `SoAdTxPduCollectionSemantics` set to `SOAD_COLLECT_LAST_IS_BEST`, SoAd shall only transmit the last instance of each PDU with the same PduId in the sequence their first instances were requested for transmission.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00736] SoAd shall reject configurations in which the transmit properties (see `SoAdTxPduCollectionSemantics`) of the PDUs which are assigned to a socket connection are mixed. Furthermore all socket connections of a socket connection group shall either be referred solely by PDUs with `SOAD_COLLECT_LAST_IS_BEST` or solely by PDUs with `SOAD_COLLECT_QUEUED` semantic.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00696] SoAd shall maintain a nPdu specific timer for each socket connection using the nPdu feature.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00550] Within `SoAd_MainFunction()` SoAd shall transmit all PDUs stored for a socket connection (if any) by calling `TcpIp_UdpTransmit()` if the nPdu specific timer expired.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00697] If a PDU with `TriggerMode` set to `TRIGGER_NEVER` with a specific `SoAdTxUdpTriggerTimeout` is transmitted SoAd shall set the nPdu specific timer to the value of `SoAdTxUdpTriggerTimeout` if the timer is not running or if it is lower than the current nPdu specific timer value. ] (SRS\_Eth\_00116)

[SWS\_SoAd\_00683] If a PDU with `TriggerMode` set to `TRIGGER_NEVER` without a specific `SoAdTxUdpTriggerTimeout` is transmitted SoAd shall set the the nPdu specific timer to the value of `SoAdSocketUdpTriggerTimeout` if the timer is not running or if it is lower than the current nPdu specific timer value.] (SRS\_Eth\_00116)

[SWS\_SoAd\_00684] SoAd shall stop the nPdu specific timer when the PDUs stored for the socket connection have been sent] (SRS\_Eth\_00116)

[SWS\_SoAd\_00737] For all PDUs that have been stored for a socket connection using the nPdu feature, SoAd shall call the upper layer with the related transmit confirmation function (`<Up>_[SoAd][If]TxConfirmation()`) within the context of the `SoAd_MainFunction()` and with result set to

- (1) `E_OK` if the related `TcpIp_UdpTransmit()` call was successful,
- (2) `E_NOT_OK` if the transmission was not successful or cancelled for any other reason.] ()

### 7.2.3 PDU Transmission via `IfRoutingGroupTransmit` API

[SWS\_SoAd\_00662] At `SoAd>IfRoutingGroupTransmit()` SoAd shall store a trigger transmit request for each `SoAdPduRouteDest` that contains a reference to the routing group identified by the parameter `id` for transmission in the `SoAd_MainFunction().]` ()

[SWS\_SoAd\_00720] At `SoAd>IfSpecificRoutingGroupTransmit()` SoAd shall store a trigger transmit request for each `SoAdPduRouteDest` that contains a reference to the routing group identified by the parameter `id` for transmission on the socket connection identified by the parameter `SoConId` in the `SoAd_MainFunction().]` ()

[SWS\_SoAd\_00665][ In the `SoAd_MainFunction()` the SoAd shall check for pending triggered transmit request for `SoAdPduRouteDest` and identify all related IF-PDUs. For each identified IF-PDU SoAd shall process as specified below:

- (1) retrieve the data from the related upper layer by calling `<Up>_[SoAd][If]-TriggerTransmit()` and
- (2) transmit the data via the related socket connection.

] ()

[SWS\_SoAd\_00728] To trigger PDU data from an upper layer SoAd shall set `PduInfoType.SduDataPtr` to the location of the buffer where the data shall be copied, set `PduInfoType.SduDataLength` to the length of this buffer and then call `<Up>_[SoAd][If]-TriggerTransmit().]` ()

### 7.2.4 PDU Transmission via TP-API

[SWS\_SoAd\_00551] For the transmission of a PDU requested by an upper layer using the TP-API, the SoAd shall

- (1) Skip further processing and return with `E_NOT_OK` if the PDU length is 0.
- (2) Identify the related socket connection and PDU route by using the `TxPduId` provided at `SoAd_TpTransmit()`.
- (3) Store the TP transmission request for further processing in the `SoAd_MainFunction()`.

] ()



Note: TxPduId identifies a SoAdPduRoute in the SoAd configuration which contains one or more SoAdPduRouteDest container which references to a SoAdSocketConnection

[SWS\_SoAd\_00552] In the SoAd\_MainFunction() the SoAd shall check for pending TP transmission requests and process a pending request as specified below:

- (1) Query the available amount of data at the upper layer by calling the configurable callback function `<Up>_[SoAd][Tp]CopyTxData()` with `PduInfoType.SduLength = 0`.
- (2) Depending on the connection type: retrieve data and call the related Tcplp transmit function.

] ()

[SWS\_SoAd\_00553] In case of a UDP socket connection the SoAd shall

- (1) retrieve all available data from the upper layer to a SoAd TP transmit buffer via the configurable callback function `<Up>_[SoAd][Tp]CopyTxData()` with `PduInfoType.SduLength` set to the value returned by `availableDataPtr` of the previous call and
- (2) call `TcpIp_UdpTransmit()` with `SocketId` and remote address specified in the `SocketConnection` and the PDU length specified in the `SoAd_TpTransmit()` call as `TotalLength` after all data have been successfully retrieved within one or multiple SoAd main function execution cycles.

] ()

Note: The required TP buffer size for a socket connection can be derived from the length of the related TP PDU(s).

[SWS\_SoAd\_00652] If `<Up>_[SoAd][Tp]CopyTxData()` return with `BUFREQ_E_NOT_OK` for a UDP socket connection, SoAd shall immediately terminate the TP transmit session and notify the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation()`) with `E_NOT_OK` as result. (Note: the related socket connection is not closed in this case.)] ()

[SWS\_SoAd\_00554] In case of a TCP socket connection the SoAd shall call `Tcplp_TcpTransmit()` with `SocketId` specified in the `SocketConnection`, the PDU length set to the value returned by `availableDataPtr` of the previous call to `<Up>_[SoAd][Tp]CopyTxData()` as `AvailableLength` and `ForceRetrieve` set to `FALSE`.] ()

The Tcplp module will retrieve PDU data from SoAd within the context of the Tcplp transmit call by using `SoAd_CopyTxData()`.

[SWS\_SoAd\_00555] In case of a UDP socket connection the SoAd shall copy (the requested part of) the PDU from the SoAd TP transmit buffer to the memory specified by parameter `BufPtr` within `SoAd_CopyTxData()`.

] ()

[SWS\_SoAd\_00556] In case of a TCP socket connection the SoAd shall forward the request to the related upper layer by calling `<Up>_[SoAd][Tp]CopyTxData()` to copy (the requested part of) the PDU to the memory specified by parameter `BufPtr` within `SoAd_CopyTxData()`.

] ()

[SWS\_SoAd\_00651] If `<Up>_[SoAd][Tp]CopyTxData()` return with `BUFREQ_E_NOT_OK` for a TCP socket connection, SoAd shall (a) disable further transmission or reception for this socket connection (i.e. new transmit requests shall be rejected with `E_NOT_OK` and received messages shall simply be discarded) and (b) close the socket connection in the next `SoAd_MainFunction()`.] ()

[SWS\_SoAd\_00557] In case of a UDP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation()`) and `E_OK` as result within the `SoAd_MainFunction()` after `TcpIp_UdpTransmit()` returns with `TCPIP_OK`.] ()

[SWS\_SoAd\_00667] In case of a TCP socket connection configured with `SoAdSocketTcplImmediateTpTxConfirmation` set to `TRUE` the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation()`) and `E_OK` as result within the `SoAd_MainFunction()` after `TcpIp_TcpTransmit()` returns `E_OK`.] ()

[SWS\_SoAd\_00670] In case of a TCP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation()`) and `E_NOT_OK` as result within the `SoAd_MainFunction()` after `TcpIp_TcpTransmit()` returns with `E_NOT_OK`.] ()

[SWS\_SoAd\_00558] In case of a TCP socket connection configured with `SoAdSocketTcplImmediateTpTxConfirmation` set to `FALSE` the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation()`) and `E_OK` as result within the `SoAd_TxConfirmation()` callback function after all TP PDU data have been confirmed for transmission.] ()

Note: `SoAd_TpTransmit()` for a new TP session with the same PDU can be called within `<Up>_[SoAd][Tp]TxConfirmation()`.

### 7.3 PDU Header option

[SWS\_SoAd\_00197] In case PDU header option is enabled (`SoAdPduHeader-Enable` is `TRUE`) for a socket connection and PDU transmission, SoAd shall insert the PDU Header with the configured `HeaderId` and the actual PDU length directly

before the PDU data, i.e. `TcpIp_UdpTransmit()` or `TcpIp_TcpTransmit()` shall be called with a `TotalLength` or `AvailableLength` increased by the PDU Header length, the PDU Header shall be copied before the PDU data to a SoAd UDP transmit buffer (if any) and a memory specified by `TcpIp` within `SoAd_CopyTxData()` requesting the begin of the PDU data.

] ()

[SWS\_SoAd\_00198] [The SoAd PDU header shall consist of a 4 byte ID field for unique identification of the PDU at the receiver and a 4 byte length field specifying the data length of the PDU. Both in BigEndian byte order.] ()

## 7.4 PDU Reception

For the reception of a PDU via an UDP or TCP socket, the SoAd configuration specifies a socket route which refers to a socket connection. A socket route (`SoAdSocketRoute`, `SoAdSocketRouteDest`) describes the route from an UDP or TCP socket of the `TcpIp` stack (which is described by the socket connection (`SoAdSocketConnection`, `SoAdSocketConnectionGroup`)) to the related upper layer module of the SoAd.

The upper layer module of the SoAd may use the Interface (IF) API or the Transport Protocol (TP) API for PDU reception.

[SWS\_SoAd\_00562] For the reception of a message from an UDP or TCP socket and forwarding of the received data as PDU to the related upper layer the SoAd shall

- (1) Identify the related socket connection and socket routes by using the `SocketId` provided at `SoAd_RxIndication()`
- (2) Filter messages according to the message acceptance policy
- (3) Convert the message into a PDU
- (4) Skip further processing if PDU length is 0 and (`SoAdPduHeaderEnable` is `FALSE` or `SoAdRxUpperLayerType` is `TP`)
- (5) Call the upper layer type related reception functions of the configured upper layer module depending on the `SoAdRxUpperLayerType` specified in `SocketRouteDest` configuration

] ()

[SWS\_SoAd\_00657] In case more than one socket connection belongs to a UDP socket connection group, a UDP socket is shared between all socket connections of the group and the related socket connection shall be selected according to the best match algorithm (see [SWS\_SoAd\_00680]). ()

[SWS\_SoAd\_00563] In case PDU header option is disabled (`SoAdPduHeaderEnable` is `FALSE`) for a socket connection, SoAd shall convert the received UDP or TCP message 1:1 into a PDU within `SoAd_RxIndication()`, i.e. each TCP segment and UDP message forms a separate PDU.] ()

[SWS\_SoAd\_00709] If `SoAdSocketUdpStrictHeaderLenCheckEnabled` is enabled SoAd shall check if the length of the received UDP message does match the

accumulated length of all PDUs including their PDU headers prior to forwarding any data to an upper layer. If the their lengths are different SoAd shall silently drop the whole message without forwarding any data.] ()

[SWS\_SoAd\_00559] In case PDU header option is enabled (`SoAdPduHeaderEnable` is `TRUE`) for a socket connection, SoAd shall convert the message into a PDU within `SoAd_RxIndication()` according to the following:

- (1) assemble the PDU Header into a SoAd receive buffer if it is fragmented in multiple TCP segments
- (2) extract the PDU Header from the received message
- (3) select the related socket route according to the received PDU Header ID (`SoAdRxPduHeaderId`); if no socket route can be found, simply discard the PDU and if development error detection is enabled: raise the error `SOAD_E_INV_PDUHEADER_ID`.
- (4) use the length field of the PDU Header to identify the length of the actual PDU and the start of the next PDU to proceed with (2) until the end of the message is reached. If the remainder is smaller than a PDU Header or the indicated length within the header SoAd shall stop processing and ignore the rest of the message.

] ()

[SWS\_SoAd\_00710] In case no valid PDU data was forwarded to an upper layer and the remote address of the socket connection was overwritten according to [SWS\_SoAd\_00592] in context of the same `SoAd_RxIndication()`, SoAd shall revert the remote address change and set the state of the socket connection back to `SOAD_SOCON_RECONNECT`.] ()

[SWS\_SoAd\_00564] In case of a TCP socket connection the SoAd shall confirm the reception of all data either forwarded to the upper layer or finally handled by SoAd (e.g. discarded data or processed PDU Header) by calling `TcpIp_TcpReceived()` within `SoAd_RxIndication()` or `SoAd_MainFunction()` respectively.] ()

[SWS\_SoAd\_00565] SoAd shall process TP- and IF-PDUs independently and within each type according to the received order per socket connection.] ()

Note: an ongoing TP reception on a socket connection blocks further TP receptions on the same socket connection, but does not block any reception of IF-PDUs.

[SWS\_SoAd\_00566] SoAd shall preserve the order of received data when using a SoAd receive buffer ] ().

[SWS\_SoAd\_00693] If development error detection is enabled: Whenever a PDU or a part of a PDU is received, that has to be stored in a SoAd receive buffer, is larger than the remaining available buffer size SoAd shall raise the development error `SOAD_E_NOBUFS`.] ()

[SWS\_SoAd\_00758] [ If the measurement data is enabled (see SoAdGetAndResetMeasurementDataApi), SoAd shall increment the corresponding measurement data whenever a received PDU is discarded.] (SRS\_Eth\_00131)

#### 7.4.1 PDU Reception via IF-API

[SWS\_SoAd\_00567] SoAd shall perform the following further actions within the `SoAd_RxIndication()` function for reception of a PDU to an upper layer using the IF-API:

- (1) assemble all data of a fragmented IF-PDU into a SoAd receive buffer if PDU Header is used
  - (2) call `<Up>_[SoAd][If]RxIndication()` of the related upper layer module (with `RxPduId` set to the ID specified by the upper layer module for the PDU referenced by `SoAdRxPduRef`) for each completely received PDU
  - (3) dispatch the next IF-PDU (if any) if PDU Header mode is used
- ] ()

Note: IF-PDU fragmentation is only supported for TCP socket connections with PDU Header mode enabled as UDP does not guarantee the message order and TCP segments are considered as separate PDUs if PDU Header mode is disabled.

[SWS\_SoAd\_00740] In case of a reception related to a `SoAdSocketRouteDest`, that refers to a global PDU structure configured with a `MetaDataItem` of the type `SOCKET_CONNECTION_ID_16`, which is contained in a `SoAdSocketRoute` that refers to a socket connection group, SoAd shall use `PduInfoType.MetaDataPtr` to provide the socket connection identifier (`SoConId`) where the PDU was received with `<Up>_[SoAd][If]RxIndication().]` (SRS\_Eth\_00124)

#### 7.4.2 PDU Reception via TP-API (PDU Header disabled)

[SWS\_SoAd\_00568] SoAd shall perform the following further actions within the `SoAd_RxIndication()` function for reception of a PDU from a socket connection with PDU Header mode disabled to an upper layer using the TP-API:

- (1) if the SoAd receive buffer does not contain any TP data for this socket connection
  - (a) Query the available amount of data at the upper layer by calling the configurable callback function `<Up>_[SoAd][Tp]CopyRxData()` with `PduInfoType.SduLength = 0`.
  - (b) If not all data can be processed (i.e. forwarded to an upper layer or stored in a SoAd receive buffer), discard all received data and skip further processing.
  - (c) Copy all received data which can be accepted by the upper layer module determined at (a) to the upper layer by calling `<Up>_[SoAd][Tp]CopyRxData()`

- (d) Copy all remaining data (i.e. data which are received but not delivered to the upper layer) to a SoAd receive buffer for later processing by `SoAd_MainFunction()`
  - (2) if the SoAd receive buffer already contains TP data for this socket connection and is able to store all (newly) received data: copy all received data to the SoAd receive buffer for later processing by `SoAd_MainFunction()`
- ] ()

[SWS\_SoAd\_00569] In the `SoAd_MainFunction()` the SoAd shall process as specified below if the SoAd receive buffer contains TP data for a socket connection with PDU Header mode disabled:

- (1) Query the available amount of data at the upper layer by calling the configurable callback function `<Up>_[SoAd][Tp]CopyRxData()` with `PduInfoType.SduLength = 0`.
  - (2) Copy all data belonging to this socket connection from the SoAd receive buffer which can be accepted by the upper layer module determined at (1) to the upper layer by calling `<Up>_[SoAd][Tp]CopyRxData()`
- ] ()

[SWS\_SoAd\_00570] If `<Up>_[SoAd][Tp]StartOfReception()` or `<Up>_[SoAd][Tp]CopyRxData()` return with `BUFREQ_E_NOT_OK` for a socket connection with PDU Header mode disabled, SoAd shall (a) disable further transmission or reception for this socket connection (i.e. new transmit requests shall be rejected with `E_NOT_OK` and received messages shall simply be discarded) and (b) close the socket connection in the next `SoAd_MainFunction()`.] ()

Note: SoAd will call `<User_SoAdTp>RxIndication` with `E_NOT_OK` in case the socket connection is disconnected while an active TP reception.

### 7.4.3 PDU Reception via TP-API (PDU Header enabled)

[SWS\_SoAd\_00571] SoAd shall perform the following further actions within the `SoAd_RxIndication()` function for reception of a PDU from a socket connection with PDU Header mode enabled to an upper layer using the TP-API:

- (1) if no TP reception is in progress for the related socket connection
  - (a) After reception of a complete PDU Header, call `<Up>_[SoAd][Tp]StartOfReception()` of the related upper layer module with `RxPduId` set to the ID specified by the upper layer module for the PDU referenced by `SoAdRxPduRef`, set `TpSduLength` to the length specified in the PDU Header, and set `PduInfoType.SduDataPtr` and `PduInfoType.SduLength` to provide already received PDU data to the upper layer.
  - (b) if not all data can be processed (i.e. forwarded to an upper layer or stored in a SoAd receive buffer), discard all received data, call `<Up>_[SoAd][Tp]RxIndication()` with the same `RxPduId` as used at `<Up>_[SoAd][Tp]StartOfReception()` and result set to `E_NOT_OK` and skip further processing



- (c) call `<Up>_[SoAd][Tp]CopyRxData()` of the related upper layer module with the same `RxPduId` as used at `<Up>_[SoAd][Tp]StartOfReception()` and `PduInfoType.SduDataPtr` pointing to the PDU data provided by `SoAd_RxIndication()` and `PduInfoType.SduLength` set to minimum of the received PDU data and the available receive buffer in the upper layer module specified by `bufferSizePtr` at `<Up>_[SoAd][Tp]-StartOfReception()`
  - (d) call `<Up>_[SoAd][Tp]RxIndication()` if the complete PDU has been forwarded to the upper layer, otherwise copy all received data which could not be forwarded to the upper layer to a SoAd receive buffer for later processing by `SoAd_MainFunction()`
- (2) if a TP reception is in progress for the related socket connection and the related SoAd receive buffer is able to store all received data: copy all received data to the related SoAd receive buffer for later processing by `SoAd_MainFunction()`
- ] ()

[SWS\_SoAd\_00572] If `<Up>_[SoAd][Tp]StartOfReception()` does not return `BUFREQ_OK` for a socket connection with PDU Header mode enabled, SoAd shall simply discard all data of the PDU.] ()

Note: `<Up>_[SoAd][Tp]RxIndication()` will not be called for a PDU when `<Up>_[SoAd][Tp]StartOfReception()` does not return `BUFREQ_OK`.

[SWS\_SoAd\_00573] If `<Up>_[SoAd][Tp]CopyRxData()` does not return `BUFREQ_OK` for a socket connection with PDU Header mode enabled, SoAd shall terminate the TP receive session, simply discard all data of the PDU and call `<Up>_[SoAd][Tp]RxIndication()` with `E_NOT_OK`.] ()

[SWS\_SoAd\_00574] In the `SoAd_Mainfunction()` the SoAd shall process as specified below if a TP reception is in progress for a socket connection with PDU Header mode enabled:

- (1) Query the available amount of data at the upper layer by calling the configurable callback function `<Up>_[SoAd][Tp]CopyRxData()` with `PduInfoType.SduLength = 0`.
  - (2) Copy all data belonging to this socket connection from the SoAd receive buffer which can be accepted by the upper layer module determined at (1) to the upper layer by calling `<Up>_[SoAd][Tp]CopyRxData()`
  - (3) call `<Up>_[SoAd][Tp]RxIndication()` if the complete PDU has been forwarded to the upper layer and dispatch the next TP-PDU (if any)
- ] ()

## 7.5 Best Match Algorithm

[SWS\_SoAd\_00680] SoAd shall use the following best match algorithm to select a socket connection of a socket connection group based on a provided (specific) remote address:

- (1) socket connections that have no (specific or wildcard) remote address set shall be ignored
- (2) the remote address of the remaining socket connections shall be compared with the provided remote address and the socket connection with the best match according to the following ordered list (item listed earlier has higher priority towards items listed later) shall be selected:
  - (a) IP address and port match
  - (b) IP address match (and wildcard set for port)
  - (c) Port match (and wildcard set for IP address)
  - (d) Wildcards are used for both IP address and port
  - (e) No match (i.e. no socket connections can be selected)

] ()

## 7.6 Message Acceptance Policy

[SWS\_SoAd\_00524] If `SoAdSocketMsgAcceptanceFilterEnabled` is `TRUE`, `SoAd` shall only accept TCP connections or UDP datagrams from remote nodes with a source address that matches the remote address specified in the socket connection (either via configuration parameters `SoAdSocketRemoteIpAddress` and `SoAdSocketRemotePort` or set online with `SoAd_SetRemoteAddr()` API.) ] ()

Note: If `SoAdSocketMsgAcceptanceFilterEnabled` is `TRUE` and the remote address is not specified by the configuration or not yet set via `SoAd_SetRemoteAddr()` no message is accepted via the socket connection.

[SWS\_SoAd\_00525] A remote address matches if both IP address and port match. The IP addresses match if they are identical or if the specified IP address is set to `TCPIP_IPADDR_ANY` (`TCPIP_IP6ADDR_ANY`). The port matches if they are identical or if the specified port is set to `TCPIP_PORT_ANY`.] ()

[SWS\_SoAd\_00582] For a UDP socket connection of type automatic (i.e. configuration parameter `SoAdSocketAutomaticSoConSetup` set to `TRUE`) which uses a wildcard in the configured remote address (i.e. an ANY-String for IP address or port), `SoAd` shall (a) change the state of the socket connection to `SOAD_SOCON_RECONNECT` and (b) reset the remote address to the configured remote address after a PDU transmission, directly before the related transmit confirmation function is called (or would be called if such a function is not configured).] ()

[SWS\_SoAd\_00644] For a TCP socket connection of type automatic (i.e. configuration parameter `SoAdSocketAutomaticSoConSetup` set to `TRUE`) which uses a wildcard in the configured remote address (i.e. an ANY-String for IP address or port), `SoAd` shall (a) disable further transmission or reception for this socket connection (i.e. new transmit requests shall be rejected with `E_NOT_OK` and received messages shall simply be discarded) after a PDU transmission, directly before the related transmit confirmation function is called (or would be called if such a function is not configured) and (b) close the socket connection in the next `SoAd_MainFunction()`.] ()



[SWS\_SoAd\_00527] SoAd shall reset the remote address to the configured remote address (or unset the remote address in case no remote address has been configured) within `SoAd_MainFunction()` when a socket connection is closed.] ()

[SWS\_SoAd\_00635] If `SoAdSocketMsgAcceptanceFilterEnabled` is `FALSE`, SoAd shall accept all TCP connection or UDP datagrams from remote nodes.] ()

## 7.7 TP PDU Cancellation

[SWS\_SoAd\_00575] SoAd shall store a cancellation request when called with `SoAd_TpCancelReceive()` and `SoAd_TpCancelTransmit()`, but handle the request only in the `SoAd_MainFunction()` respecting the connection loss and recovery policy. ] ()

[SWS\_SoAd\_00576] If `SoAd_TpCancelReceive()` or `SoAd_TpCancelTransmit()` is called for a PDU where TP reception or TP transmission is not in progress, SoAd shall ignore the request and return `E_NOT_OK`.] ()

## 7.8 Disconnection and recovery

[SWS\_SoAd\_00577] Within `SoAd_MainFunction()`, SoAd shall close the socket connection for any communication cancelation request related to an active TP transmission.] ()

[SWS\_SoAd\_00581] Within `SoAd_MainFunction()`, SoAd shall close the socket connection for any communication cancelation request related to an active TP reception.] ()

[SWS\_SoAd\_00586] SoAd shall automatically reestablish a socket connection which is in the connection state `SOAD_SOCON_RECONNECT` within `SoAd_MainFunction()` - independent of the configuration parameter `SoAdSocketAutomaticSoConSetup`, i.e. connection shall be reestablished even if the parameter is set to `FALSE`. Reconnection shall be done by considering configuration parameter `SoAdSocketTcpInitiate`.] ()

[SWS\_SoAd\_00587] SoAd shall return `E_NOT_OK` for TP-PDU Tx requests received at `SoAd_TpTransmit()` within connection reestablishment.] ()

[SWS\_SoAd\_00694] If a UDP socket connection is configured with a `SoAdSocketUdpAliveSupervisionTimeout` and the remote address was overwritten, as described in [SWS\_SoAd\_00592], the alive supervision timer for this socket connection shall be started with the value specified by the configuration parameter `SoAdSocketUdpAliveSupervisionTimeout`.] (SRS\_Eth\_00085)

[SWS\_SoAd\_00742] [ If a UDP socket connection is configured with a SoAdSocketUdpAliveSupervisionTimeout and a datagram is received that passes the message acceptance filter, the timer shall be restarted with the value specified by the configuration parameter SoAdSocketUdpAliveSupervisionTimeout.] (SRS\_Eth\_00085)

[SWS\_SoAd\_00695] If a UDP socket connection is configured with a SoAdSocketUdpAliveSupervisionTimeout, the alive supervision timer runs out and the remote address was not set by the upper layer, SoAd shall

- (a) change the state of the socket connection to SOAD\_SOCON\_RECONNECT,
- (b) deactivate the alive supervision timer and
- (c) reset the remote address to the configured remote address at SoAd\_MainFunction().] (SRS\_Eth\_00085)

## 7.9 Routing Groups

To selectively enable/disable the routing of PDUs from or to socket connections routing groups are defined and can be controlled by the upper layer of the SoAd.

[SWS\_SoAd\_00601] SoAd shall maintain the state of each configured routing group and activate or deactivate the state at initialization depending on the configuration parameter SoAdRoutingGroupsEnabledAtInit.] ()

[SWS\_SoAd\_00721] For RoutingGroups that are referenced by a SoAdPduRouteDest that refers to a SocketConnectionGroup SoAd shall maintain independent states for each SocketConnection that is part of the referenced SocketConnectionGroup and handle them as if they were separate RoutingGroups.] ()

[SWS\_SoAd\_00560] If SoAd\_IfTransmit() is called with TxPduId specifying a SoAdPduRouteDest which belongs only to inactive RoutingGroups, SoAd shall always skip the transmission for this SoAdPduRouteDest and shall consider the transmission as successful unless all SoAdPduRouteDest of a SoAdPduRoute belong only to inactive RoutingGroups. In the latter case SoAd shall return E\_NOT\_OK.] ()

[SWS\_SoAd\_00561] If SoAd\_TpTransmit() is called with TxPduId specifying a SoAdPduRouteDest which belongs only to inactive RoutingGroups, SoAd shall always skip the transmission for this SoAdPduRouteDest and shall consider the transmission as successful unless all SoAdPduRouteDest of a SoAdPduRoute belong only to inactive RoutingGroups. In the latter case SoAd shall return E\_NOT\_OK.] ()

[SWS\_SoAd\_00600] If a PDU is received according to SoAdSocketRouteDest which belongs only to inactive RoutingGroups, SoAd shall simply discard the PDU.] ()

Note: activation/deactivation of a routing group only affects new PDUs, i.e. PDUs which are already in an active reception or transmission process by an upper layer (e.g. long TP-PDU which is received via a multiple CopyRxData calls) are not affected.

## 7.10 PDU fan-out

[SWS\_SoAd\_00602] SoAd shall support more than one SoAdPduRouteDest per SoAdPduRoute for upper layers of If-type, i.e. a single IF-PDU can be transmitted via multiple socket connections.] ()

[SWS\_SoAd\_00722] SoAd shall handle SoAdPduRoutes with SoAdPduRouteDests referring to a SocketConnectionGroup as if they were separate SoAdPduRouteDests referring to each SocketConnection of this Group.] ()

[SWS\_SoAd\_00648] If a transmit request on any of multiple socket connections returns E\_NOT\_OK, SoAd shall return E\_NOT\_OK at SoAd\_IfTransmit().] ()

[SWS\_SoAd\_00647] In case of multiple socket connections, SoAd shall call the upper layer with the configured transmit confirmation function (<Up>\_[SoAd][If]TxConfirmation()) with result set to E\_OK only once after transmission on all related socket connections succeeded.] ()

## 7.11 Buffer handling

[SWS\_SoAd\_00505] The SoAd shall provide sufficient buffers to store received data which can't be forwarded to the upper layer within the context of SoAd\_RxIndication as well as buffers for data which can (or should) not be forwarded to TcpIp.] ()

[SWS\_SoAd\_00599] The SoAd shall provide sufficient buffers to store data which temporarily can't be forwarded to TcpIp, e.g. SoAd UDP TP transmit buffers or UdpTxBuffer, nPduUdpTxBuffer.] ()

## 7.12 Error classification

This section describes how the SoAd module has to manage the error classes that may occur during the life cycle of this basic software.

For further details regarding the error classification please refer to Chapter 7.2. “Error Handling” in the *General Specification of Basic Software Modules* [16].

### 7.12.1 Development Errors

[SWS\_SoAd\_00101] [ The following table lists development error IDs the SoAd shall use for reporting of development errors to the Default Error Tracer:

Type or error	Relevance	Related error code	Value
API service called before initializing the module	Development	SOAD_E_NOTINIT	0x01
API service called with NULL pointer	Development	SOAD_E_PARAM_POINTER	0x02
Invalid argument	Development	SOAD_E_INV_ARG	0x03
No buffer space available	Development	SOAD_E_NOBUFS	0x04
Unknown PduHeader ID	Development	SOAD_E_INV_PDUHEADER_ID	0x05
Invalid PDU ID	Development	SOAD_E_INV_PDUID	0x06
Invalid socket address	Development	SOAD_E_INV_SOCKETID	0x07
Invalid configuration set selection	Development	SOAD_E_INIT_FAILED	0x08
Invalid meta data	Development	SOAD_E_INV_METADATA	0x09

] ()

### 7.12.2 Runtime Errors

< There are no runtime errors.>

### 7.12.3 Transient Faults

< There are no transient faults.>

### 7.12.4 Production Errors

< There are no production errors.>

### 7.12.5 Extended Production Errors

< There are no extended production errors.>

## **7.13 Application notes**

### **7.14 Debugging Concept**

For details refer to the chapter 7.1.17 “Debugging support” in *SWS\_BSWGeneral*.

### **7.15 Version checking**

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

## 8 API specification

### 8.1 Imported types

The following types shall be imported by the SoAd from the modules given:

[SWS\_SoAd\_00503] [

Module	Imported Type
ComStack_Types	BufReq_ReturnType
	PduIdType
	PduInfoType
	PduLengthType
	RetryInfoType
	TPParameterType
Std_Types	Std_ReturnType
	Std_VersionInfoType
Tcplp	Tcplp_DomainType
	Tcplp_EventType
	Tcplp_IpAddrAssignmentType
	Tcplp_IpAddrStateType
	Tcplp_LocalAddrIdType
	Tcplp_ParamIdType
	Tcplp_ProtocolType
	Tcplp_ReturnType
	Tcplp_SockAddrType
	Tcplp_SocketIdType
	Tcplp_StateType

] ()

### 8.2 Type definitions

#### 8.2.1 SoAd\_SoConIdType

[SWS\_SoAd\_00518] [

<b>Name:</b>	SoAd_SoConIdType
<b>Type:</b>	uint8, uint16
<b>Range:</b>	0..<SoAdSoConMax>--Zero-based integer number
<b>Description:</b>	SoCon identifier type for unique identification of a SoAd socket connection. The size of this type depends on the maximum number of socket connections which is specified by configuration parameter SoAdSoConMax.

] ()

#### 8.2.2 SoAd\_SoConModeType

[SWS\_SoAd\_00512] [

<b>Name:</b>	SoAd_SoConModeType
<b>Type:</b>	Enumeration
<b>Range:</b>	SOAD_SOCON_ONLINE--
	SOAD_SOCON_RECONNECT--

	SOAD_SOCON_OFFLINE	--	--
<b>Description:</b>	type to specify the state of a SoAd socket connection.		

] ()

### 8.2.3 SoAd\_RoutingGroupIdType

[SWS\_SoAd\_00519] [

<b>Name:</b>	SoAd_RoutingGroupIdType		
<b>Type:</b>	uint8, uint16		
<b>Range:</b>	0..<SoAdRoutingGroupMax>	--	Zero-based integer number
<b>Description:</b>	RoutingGroup identifier type for unique identification of a SoAd routing group. The size of this type depends on the maximum number of routing groups which is specified by configuration parameter SoAdRoutingGroupMax.		

] ()

### 8.2.4 SoAd\_ConfigType

[SWS\_SoAd\_00210] [

<b>Name:</b>	SoAd_ConfigType		
<b>Type:</b>	Structure		
<b>Range:</b>	implementation specific	The content of the configuration data structure is implementation specific.	
<b>Description:</b>	Configuration data structure of the SoAd module.		

] ()

### 8.2.5 SoAd\_MeasurementIdxType

[SWS\_SoAd\_91010] [

<b>Name:</b>	SoAd_MeasurementIdxType		
<b>Type:</b>	uint8		
<b>Range:</b>	SOAD_MEAS_DROP_TCP	0x01	Measurement index of dropped PDUs caused by invalid destination TCP-Port
	SOAD_MEAS_DROP_UDP	0x02	Measurement index of dropped PDUs caused by invalid destination UDP-Port
	SOAD_MEAS_RESERVED_1	0x03–0x7F	reserved by AUTOSAR
	SOAD_MEAS_RESERVED_2	0x80–0xEF	Vendor specific range
	SOAD_MEAS_RESERVED_3	0xF0–0xFE	reserved by AUTOSAR (future use)
	SOAD_MEAS_ALL	0xFF	represents all measurement indexes
<b>Description:</b>	Index to select specific measurement data		

] ()

## 8.3 Function definitions

### 8.3.1 General

#### 8.3.1.1 SoAd\_GetVersionInfo

##### [SWS\_SoAd\_00096] [

<b>Service name:</b>	SoAd_GetVersionInfo
<b>Syntax:</b>	void SoAd_GetVersionInfo( Std_VersionInfoType* versioninfo )
<b>Service ID[hex]:</b>	0x02
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	versioninfo   Pointer to where to store the version information of this module.
<b>Return value:</b>	None
<b>Description:</b>	Returns the version information.

] ()

#### 8.3.1.2 SoAd\_Init

##### [SWS\_SoAd\_00093] [

<b>Service name:</b>	SoAd_Init
<b>Syntax:</b>	void SoAd_Init( const SoAd_ConfigType* SoAdConfigPtr )
<b>Service ID[hex]:</b>	0x01
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	SoAdConfigPtr   Pointer to the configuration data of the SoAd module.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	Initializes the Socket Adaptor.

] ()

##### [SWS\_SoAd\_00211] [

SoAd\_Init shall store the access to the configuration structure for subsequent API calls. ] ()

[SWS\_SoAd\_00723] SoAd\_Init() initializes all global variables of a Socket Adaptor instance and puts all socket connections into the state SOAD\_SOCON\_OFFLINE.] ()



#### [SWS\_SoAd\_00216] [

If development error detection is enabled: `SoAd_Init()` shall check the parameter `SoAdConfigPtr` for containing a valid configuration. If the check fails, `SoAd_Init()` shall raise the development error `SOAD_E_INIT_FAILED`. ] ( )

### 8.3.2 Normal Operation

#### 8.3.2.1 SoAd\_IfTransmit

##### [SWS\_SoAd\_00091] [

<b>Service name:</b>	SoAd_IfTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_IfTransmit(     PduIdType TxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x49	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Transmit request has been accepted.
		E_NOT_OK: Transmit request has not been accepted.
<b>Description:</b>	Requests transmission of a PDU.	

] ( )

##### [SWS\_SoAd\_00213] [

If development error detection is enabled: `SoAd_IfTransmit()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_IfTransmit()` shall raise the development error `SOAD_E_NOTINIT`. ] ( )

##### [SWS\_SoAd\_00214] [

If development error detection is enabled: `SoAd_IfTransmit()` shall check parameter `TxPduId` for being valid. If the check fails, `SoAd_IfTransmit()` shall raise the development error `SOAD_E_INV_PDUID`. ] ( )

[SWS\_SoAd\_00732] [ SoAd shall only consider `PduInfoPtr->SduDataPtr` set to `NULL_PTR` as valid if `SoAdIfTriggerTransmit` is set to `TRUE` for the respective upper layer. ] (SRS\_BSW\_00337)

[SWS\_SoAd\_00653] [ The service `SoAd_IfTransmit()` shall skip the transmit request and return `E_NOT_OK` if there is already an IF or TP transmission ongoing on the related socket connection identified by `TxPduId`. ] ( )

Note: An IF transmission is considered as ongoing until SoAd\_IfTransmit returns. A TP transmission is considered as ongoing until SoAd calls <Up>\_[SoAd][Tp]TxConfirmation.

### 8.3.2.2 SoAd\_IfRoutingGroupTransmit

[SWS\_SoAd\_00656] [

<b>Service name:</b>	SoAd_IfRoutingGroupTransmit	
<b>Syntax:</b>	Std_ReturnType SoAd_IfRoutingGroupTransmit( SoAd_RoutingGroupIdType id )	
<b>Service ID[hex]:</b>	0x1D	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	routing group identifier indirectly specifying PDUs to be transmitted (after requesting the newest data from the related upper layer).
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	Triggers the transmission of all If-TxPDUs identified by the parameter id after requesting the data from the related upper layer.	

] ()

[SWS\_SoAd\_00661] [ If development error detection is enabled: SoAd\_IfRoutingGroupTransmit() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_IfRoutingGroupTransmit() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00658] [ If development error detection is enabled: SoAd\_IfRoutingGroupTransmit() shall check parameter id for being valid (i.e. id refers a routing group that has configuration parameter SoAdRoutingGroupTxTriggerable set to TRUE). If the check fails, SoAd\_IfRoutingGroupTransmit() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.2.3 SoAd\_IfSpecificRoutingGroupTransmit

[SWS\_SoAd\_00711] [

<b>Service name:</b>	SoAd_IfSpecificRoutingGroupTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_IfSpecificRoutingGroupTransmit(     SoAd_RoutingGroupIdType id,     SoAd_SoConIdType SoConId )</pre>	
<b>Service ID[hex]:</b>	0x1f	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	routing group identifier indirectly specifying PDUs to be transmitted (after requesting the newest data from the related upper layer).
	SoConId	socket connection index specifying the socket connection on which the PDUs shall be transmitted
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK The request was successful. E_NOT_OK The request was not successful.
<b>Description:</b>	Triggers the transmission of all If-TxPDUs identified by the parameter id on the socket connection specified by SoConId after requesting the data from the related upper layer.	

] ()

[SWS\_SoAd\_00712] If development error detection is enabled: SoAd\_IfSpecificRoutingGroupTransmit() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_IfSpecificRoutingGroupTransmit() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00713] If development error detection is enabled: SoAd\_IfSpecificRoutingGroupTransmit() shall check parameter id for being valid (i.e. id refers a routing group that has configuration parameter SoAdRoutingGroupTxTriggerable set to TRUE). If the check fails, SoAd\_IfSpecificRoutingGroupTransmit() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.2.4 SoAd\_TpTransmit

[SWS\_SoAd\_00105] [

<b>Service name:</b>	SoAd_TpTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_TpTransmit(     PduIdType TxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x49	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.

<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
<b>Description:</b>	Requests transmission of a PDU.	

] ()

#### [SWS\_SoAd\_00224] [

If development error detection is enabled: SoAd\_TpTransmit() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_TpTransmit() shall raise the development error SOAD\_E\_NOTINIT. ] ()

#### [SWS\_SoAd\_00237] [

If development error detection is enabled: SoAd\_TpTransmit() shall check parameter TxPduId for being valid. If the check fails, SoAd\_TpTransmit() shall raise the development error SOAD\_E\_INV\_PDUID. ] ()

[SWS\_SoAd\_00650] The service SoAd\_TpTransmit() shall skip the transmit request and return E\_NOT\_OK if there is already an IF or TP transmission ongoing on the related socket connection identified by TxPduld.] ()

Note: No TxConfirmation is required when SoAd\_TpTransmit() failed.

### 8.3.3 Transmit/Receive Cancelation API

#### 8.3.3.1 SoAd\_TpCancelTransmit

#### [SWS\_SoAd\_00522] [

<b>Service name:</b>	SoAd_TpCancelTransmit	
<b>Syntax:</b>	Std_ReturnType SoAd_TpCancelTransmit( PduIdType TxPduId )	
<b>Service ID[hex]:</b>	0x4a	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pduals. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	TxPduld	Identification of the PDU to be cancelled.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination module.
<b>Description:</b>	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	

] ()

[SWS\_SoAd\_00605] If development error detection is enabled: SoAd\_TpCancelTransmit() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_TpCancelTransmit() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00606] If development error detection is enabled: SoAd\_TpCancelTransmit() shall check parameter TxPduId for being valid. If the check fails, SoAd\_TpCancelTransmit() shall raise the development error SOAD\_E\_INV\_PDUID.] ()

### 8.3.3.2 SoAd\_TpCancelReceive

[SWS\_SoAd\_00521] [

<b>Service name:</b>	SoAd_TpCancelReceive	
<b>Syntax:</b>	Std_ReturnType SoAd_TpCancelReceive( PduIdType RxPduId )	
<b>Service ID[hex]:</b>	0x4c	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	RxPduId	Identification of the PDU to be cancelled.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination module.
<b>Description:</b>	Requests cancellation of an ongoing reception of a PDU in a lower layer transport protocol module.	

] ()

[SWS\_SoAd\_00607] If development error detection is enabled: SoAd\_TpCancelReceive() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_TpCancelReceive() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00608] If development error detection is enabled: SoAd\_TpCancelReceive() shall check parameter RxPduId for being valid. If the check fails, SoAd\_TpCancelReceive() shall raise the development error SOAD\_E\_INV\_PDUID.] ()

## 8.3.4 Information and Control API

### 8.3.4.1 SoAd\_GetSoConId

[SWS\_SoAd\_00509] [

<b>Service name:</b>	SoAd_GetSoConId	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_GetSoConId(     PduIdType TxPduId,     SoAd_SoConIdType* SoConIdPtr )</pre>	
<b>Service ID[hex]:</b>	0x07	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	TxPduId	Transmit PduId specifying the SoAd socket connection for which the socket connection index shall be returned.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	SoConIdPtr	Pointer to memory receiving the socket connection index asked for.
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful
<b>Description:</b>	Returns socket connection index related to the specified TxPduId.	

] ()

[SWS\_SoAd\_00609] If development error detection is enabled: SoAd\_GetSoConId() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_GetSoConId() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00724] In case SoAd\_GetSoConId() is called with a TxPduId related to a SoAdPduRoute with a fan-out (i.e. multiple SoAdPduRouteDest specified), SoAd\_GetSoConId() shall skip further processings and return E\_NOT\_OK.] ()

[SWS\_SoAd\_00610] If development error detection is enabled: SoAd\_GetSoConId() shall check parameter TxPduId for being valid. If the check fails, SoAd\_GetSoConId() shall raise the development error SOAD\_E\_INV\_PDUID.] ()

### 8.3.4.2 SoAd\_OpenSoCon

[SWS\_SoAd\_00510] [

<b>Service name:</b>	SoAd_OpenSoCon	
<b>Syntax:</b>	Std_ReturnType SoAd_OpenSoCon( SoAd_SoConIdType SoConId )	
<b>Service ID[hex]:</b>	0x08	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection which shall be opened
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	This service opens the socket connection specified by SoConId.	

] ()

[SWS\_SoAd\_00615] If development error detection is enabled: SoAd\_OpenSoCon() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_OpenSoCon() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00611] If development error detection is enabled: SoAd\_OpenSoCon() shall check parameter SoConId for being valid. If the check fails, SoAd\_OpenSoCon() shall raise the development error SOAD\_E\_INV\_ARG.] ()

[SWS\_SoAd\_00528] If development error detection is enabled: In case SoAd\_OpenSoCon() is called for a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to "TRUE" the development error SOAD\_E\_INV\_ARG shall be raised.] ()

### 8.3.4.3 SoAd\_CloseSoCon

[SWS\_SoAd\_00511] [

<b>Service name:</b>	SoAd_CloseSoCon	
<b>Syntax:</b>	Std_ReturnType SoAd_CloseSoCon( SoAd_SoConIdType SoConId, boolean abort )	
<b>Service ID[hex]:</b>	0x09	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection which shall be closed
	abort	TRUE: socket connection will immediately be terminated. FALSE: socket connection will be terminated if no other upper layer is using this socket connection.



<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	This service closes the socket connection specified by SoConId.	

] ()

[SWS\_SoAd\_00616] If development error detection is enabled: SoAd\_CloseSoCon() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_CloseSoCon() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00612] If development error detection is enabled: SoAd\_CloseSoCon() shall check parameter SoConId for being valid. If the check fails, SoAd\_CloseSoCon() shall raise the development error SOAD\_E\_INV\_ARG.] ()

[SWS\_SoAd\_00529] If development error detection is enabled: In case SoAd\_CloseSoCon() is called for a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to "TRUE" the development error SOAD\_E\_INV\_ARG shall be raised.] ()

#### 8.3.4.4 SoAd\_GetSoConMode

[SWS\_SoAd\_91001] [

<b>Service name:</b>	SoAd_GetSoConMode	
<b>Syntax:</b>	<pre>void SoAd_GetSoConMode(     SoAd_SoConIdType SoConId,     SoAd_SoConModeType* ModePtr )</pre>	
<b>Service ID[hex]:</b>	0x22	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection for which the state shall be returned.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	ModePtr	Pointer to memory where the socket connection state shall be stored.
<b>Return value:</b>	None	
<b>Description:</b>	Returns current state of the socket connection specified by SoConId.	

] ()

#### 8.3.4.5 SoAd\_RequestIpAddrAssignment

[SWS\_SoAd\_00520] [



<b>Service name:</b>	SoAd_RequestIpAddrAssignment	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_RequestIpAddrAssignment(     SoAd_SoConIdType SoConId,     TcpIp_IpAddrAssignmentType Type,     const TcpIp_SockAddrType* LocalIpAddrPtr,     uint8 Netmask,     const TcpIp_SockAddrType* DefaultRouterPtr )</pre>	
<b>Service ID[hex]:</b>	0x0A	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in):</b>	SoConId	Socket connection index specifying the socket connection for which the IP address shall be set
	Type	Type of IP address assignment which shall be initiated.
	LocalIpAddrPtr	Pointer to structure containing the IP address which shall be assigned to the EthIf controller indirectly specified via SoConId. Note: This parameter is only used in case the parameter Type is set to TCPIP_IPADDR_ASSIGNMENT_STATIC, can be set to NULL_PTR otherwise.
	Netmask	Network mask of IPv4 address or address prefix of IPv6 address in CIDR Notation. Note: This parameter is only used in case the parameter Type is set to TCPIP_IPADDR_ASSIGNMENT_STATIC.
	DefaultRouterPtr	Pointer to structure containing the IP address of the default router (gateway) which shall be assigned. Note: This parameter is only used in case the parameter Type is set to TCPIP_IPADDR_ASSIGNMENT_STATIC, can be set to NULL_PTR otherwise.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
<b>Description:</b>	By this API service the local IP address assignment which shall be used for the socket connection specified by SoConId is initiated.	

] ()  
[SWS\_SoAd\_00613] If development error detection is enabled: SoAd\_RequestIpAddrAssignment() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_RequestIpAddrAssignment() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00617] If development error detection is enabled, SoAd\_RequestIpAddrAssignment() shall check parameter SoConId for being valid. If the check fails, SoAd\_RequestIpAddrAssignment() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.4.6 SoAd\_ReleaseIpAddrAssignment

[SWS\_SoAd\_00536] [

<b>Service name:</b>	SoAd_ReleaseIpAddrAssignment	
<b>Syntax:</b>	Std_ReturnType SoAd_ReleaseIpAddrAssignment( SoAd_SoConIdType SoConId )	
<b>Service ID[hex]:</b>	0x0B	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection for which the IP address shall be released
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
<b>Description:</b>	By this API service the local IP address assignment used for the socket connection specified by SoConId is released.	

] ()

[SWS\_SoAd\_00618] If development error detection is enabled: SoAd\_ReleaseIpAddrAssignment() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_ReleaseIpAddrAssignment() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00619] If development error detection is enabled: SoAd\_ReleaseIpAddrAssignment() shall check parameter SoConId for being valid. If the check fails, SoAd\_ReleaseIpAddrAssignment() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.4.7 SoAd\_GetLocalAddr

[SWS\_SoAd\_00506] [

<b>Service name:</b>	SoAd_GetLocalAddr	
<b>Syntax:</b>	Std_ReturnType SoAd_GetLocalAddr( SoAd_SoConIdType SoConId, TcpIp_SockAddrType* LocalAddrPtr, uint8* NetmaskPtr, TcpIp_SockAddrType* DefaultRouterPtr )	
<b>Service ID[hex]:</b>	0x0C	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index representing the SoAd socket connection for which the actual local IP address shall be obtained.
<b>Parameters (inout):</b>	LocalAddrPtr	Pointer to a struct where the local address (IP address and port) is stored. The struct member domain shall be set by the caller of the API to the desired TcpIp_DomainType and it shall be ensured by the caller that the struct is large enough to store an address of the selected type (INET or INET6).
	DefaultRouterPtr	Pointer to struct where the IP address of the default router (gateway) is stored (struct member "port" is not used and of

		arbitrary value). The struct must be of the same type and size as LocalAddrPtr.
<b>Parameters (out):</b>	NetmaskPtr	Pointer to memory where Network mask of IPv4 address or address prefix of IPv6 address in CIDR Notation is stored
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	Retrieves the local address (IP address and port) actually used for the SoAd socket connection specified by SoConId, the netmask and default router	

] ()

[SWS\_SoAd\_00621] If development error detection is enabled: SoAd\_GetLocalAddr() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_GetLocalAddr() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00620] If development error detection is enabled: SoAd\_GetLocalAddr() shall check parameter SoConId for being valid. If the check fails, SoAd\_GetLocalAddr() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.4.8 SoAd\_GetPhysAddr

[SWS\_SoAd\_00507] [

<b>Service name:</b>	SoAd_GetPhysAddr	
<b>Syntax:</b>	Std_ReturnType SoAd_GetPhysAddr( SoAd_SoConIdType SoConId, uint8* PhysAddrPtr )	
<b>Service ID[hex]:</b>	0x0D	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index representing the SoAd socket connection for which the physical source address of the related EthIf controller shall be obtained.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	PhysAddrPtr	Pointer to the memory where the physical source address (MAC address) in network byte order is stored
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	Retrieves the physical source address of the EthIf controller used by the SoAd socket connection specified by SoConId.	

] ()

[SWS\_SoAd\_00623] If development error detection is enabled: SoAd\_GetPhysAddr() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_GetPhysAddr() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00622] If development error detection is enabled: `SoAd_GetPhysAddr()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_GetPhysAddr()` shall raise the development error `SOAD_E_INV_ARG.` ()

### 8.3.4.9 SoAd\_GetRemoteAddr

[SWS\_SoAd\_00655] [

<b>Service name:</b>	SoAd_GetRemoteAddr	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_GetRemoteAddr(     SoAd_SoConIdType SoConId,     TcpIp_SockAddrType* IpAddrPtr )</pre>	
<b>Service ID[hex]:</b>	0x1C	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	SoConId	socket connection index representing the SoAd socket connection for which the actually specified remote address shall be obtained.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	IpAddrPtr	Pointer to a struct where the retrieved remote address (IP address and port) is stored.
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	Retrieves the remote address (IP address and port) actually used for the SoAd socket connection specified by <code>SoConId</code>	

] ()

[SWS\_SoAd\_00659] If development error detection is enabled: `SoAd_GetRemoteAddr()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_GetRemoteAddr()` shall raise the development error `SOAD_E_NOTINIT.` ()

[SWS\_SoAd\_00660] If development error detection is enabled: `SoAd_GetRemoteAddr()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_GetRemoteAddr()` shall raise the development error `SOAD_E_INV_ARG.` ()

[SWS\_SoAd\_00666] `SoAd_GetRemoteAddr()` shall immediately return `E_NOT_OK` if the remote address of the socket connection specified by parameter `SoConId` is not set. ()

[SWS\_SoAd\_00664] At `SoAd_GetRemoteAddr()` SoAd shall retrieve the remote address (IP address and port) actually used for the socket connection specified by parameter `SoConId.` ()

[SWS\_SoAd\_00698] `SoAd_GetRemoteAddr()` shall refuse the request if the domain set in `IpAddrPtr` does not match the `TcpIp_DomainType` of the local

address related to the socket connection identified by `SoConId` and return `E_NOT_OK`. If development error detection is enabled, the service `SoAd_GetRemoteAddr()` shall also raise the development error `SOAD_E_INV_ARG.`] ()

#### 8.3.4.10 SoAd\_EnableRouting

[SWS\_SoAd\_00516] [

<b>Service name:</b>	SoAd_EnableRouting	
<b>Syntax:</b>	Std_ReturnType SoAd_EnableRouting( SoAd_RoutingGroupIdType id )	
<b>Service ID[hex]:</b>	0x0E	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	routing group identifier specifying the routing group to be enabled
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	Enables routing of a group of PDUs in the SoAd related to the RoutingGroup specified by parameter id. Routing of PDUs can be either forwarding of PDUs from the upper layer to a TCP or UDP socket of the TCP/IP stack specified by a PduRoute or the other way around specified by a SocketRoute.	

] ()

[SWS\_SoAd\_00624] If development error detection is enabled: `SoAd_EnableRouting()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_EnableRouting()` shall raise the development error `SOAD_E_NOTINIT.`] ()

[SWS\_SoAd\_00625] If development error detection is enabled: `SoAd_EnableRouting()` shall check parameter `id` for being valid. If the check fails, `SoAd_EnableRouting()` shall raise the development error `SOAD_E_INV_ARG.`] ()

#### 8.3.4.11 SoAd\_EnableSpecificRouting

[SWS\_SoAd\_00714] [

<b>Service name:</b>	SoAd_EnableSpecificRouting	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_EnableSpecificRouting(     SoAd_RoutingGroupIdType id,     SoAd_SoConIdType SoConId )</pre>	
<b>Service ID[hex]:</b>	0x20	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	routing group identifier specifying the routing group to be enabled
	SoConId	socket connection index specifying the socket connection on which the routing group shall be enabled
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK The request was successful.
		E_NOT_OK The request was not successful.
<b>Description:</b>	Enables routing of a group of PDUs in the SoAd related to the RoutingGroup specified by parameter id only on the socket connection identified by SoConId.	

] ()

[SWS\_SoAd\_00715] If development error detection is enabled: SoAd\_EnableSpecificRouting() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_EnableSpecificRouting() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00716] If development error detection is enabled: SoAd\_EnableSpecificRouting() shall check parameter id for being valid. If the check fails, SoAd\_EnableSpecificRouting() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.4.12 SoAd\_DisableRouting

[SWS\_SoAd\_00517] [

<b>Service name:</b>	SoAd_DisableRouting	
<b>Syntax:</b>	Std_ReturnType SoAd_DisableRouting( SoAd_RoutingGroupIdType id )	
<b>Service ID[hex]:</b>	0x0F	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	routing group identifier specifying the routing group to be disabled
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
<b>Description:</b>	Disables routing of a group of PDUs in the SoAd related to the RoutingGroup specified by parameter id. Routing of PDUs can be either forwarding of PDUs from the upper layer to a TCP or UDP socket of the TCP/IP stack specified by a PduRoute or the other way around specified by a SocketRoute.	

] ()

[SWS\_SoAd\_00627] If development error detection is enabled: SoAd\_DisableRouting() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_DisableRouting() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00626] If development error detection is enabled: SoAd\_DisableRouting() shall check parameter id for being valid. If the check fails, SoAd\_DisableRouting() shall raise the development error SOAD\_E\_INV\_ARG.] ()

### 8.3.4.13 SoAd\_DisableSpecificRouting

[SWS\_SoAd\_00717] [

<b>Service name:</b>	SoAd_DisableSpecificRouting	
<b>Syntax:</b>	Std_ReturnType SoAd_DisableSpecificRouting( SoAd_RoutingGroupIdType id, SoAd_SoConIdType SoConId )	
<b>Service ID[hex]:</b>	0x21	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	routing group identifier specifying the routing group to be disabled
	SoConId	socket connection index specifying the socket connection on which the routing group shall be disabled



<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK The request was successful. E_NOT_OK The request was not successful.
<b>Description:</b>	Disables routing of a group of PDUs in the SoAd related to the RoutingGroup specified by parameter id only on the socket connection identified by SoConId.	

] ()

[SWS\_SoAd\_00718] If development error detection is enabled: SoAd\_DisableSpecificRouting() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_DisableSpecificRouting() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00719] If development error detection is enabled: SoAd\_DisableSpecificRouting() shall check parameter id for being valid. If the check fails, SoAd\_DisableSpecificRouting() shall raise the development error SOAD\_E\_INV\_ARG.] ()

#### 8.3.4.14 SoAd\_SetRemoteAddr

[SWS\_SoAd\_00515] [

<b>Service name:</b>	SoAd_SetRemoteAddr	
<b>Syntax:</b>	Std_ReturnType SoAd_SetRemoteAddr( SoAd_SoConIdType SoConId, const TcpIp_SockAddrType* RemoteAddrPtr )	
<b>Service ID[hex]:</b>	0x10	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection for which the remote address shall be set
	RemoteAddrPtr	Struct containint the IP address and port to be set.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
<b>Description:</b>	By this API service the remote address (IP address and port) of the specified socket connection shall be set.	

] ()

[SWS\_SoAd\_00628] If development error detection is enabled: SoAd\_SetRemoteAddr() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_SetRemoteAddr() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00531] If development error detection is enabled and SoConId refers to a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to TRUE, the function SoAd\_SetRemoteAddr() shall raise the development error SOAD\_E\_INV\_ARG.] ()

[SWS\_SoAd\_00532] The function `SoAd_SetRemoteAddr()` shall only proceed if `SoConId` refers to a socket connection which is not currently locked by the `SoAd`. If it is locked, the request shall be rejected and `E_NOT_OK` shall be returned.] (SRS\_Eth\_00085)

[SWS\_SoAd\_00533] The function `SoAd_SetRemoteAddr()` shall set the remote address of the socket connection referred by parameter `SoConId` according to the IP address and port specified by parameter `RemoteAddrPtr`.] ()

[SWS\_SoAd\_00687] If the function `SoAd_SetRemoteAddr()` is used to set the remote address of a socket connection that is in the mode `SOAD_SOCON_ONLINE` to a value that contains wildcards, `SoAd` shall change the mode of the socket connection to `SOAD_SOCON_RECONNECT`.] ()

[SWS\_SoAd\_00699] `SoAd_SetRemoteAddr()` shall refuse the request if the domain set in `RemoteAddrPtr` does not match the `TcpIp_DomainType` of the local address related to the socket connection identified by `SoConId` and return `E_NOT_OK`. If development error detection is enabled, the service `SoAd_SetRemoteAddr()` shall also raise the development error `SOAD_E_INV_ARG`.] ()

#### 8.3.4.15 SoAd\_SetUniqueRemoteAddr

[SWS\_SoAd\_00671] [

<b>Service name:</b>	SoAd_SetUniqueRemoteAddr	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_SetUniqueRemoteAddr(     SoAd_SoConIdType SoConId,     const TcpIp_SockAddrType* RemoteAddrPtr,     SoAd_SoConIdType* AssignedSoConIdPtr )</pre>	
<b>Service ID[hex]:</b>	0x1e	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different <code>SoConIds</code> . Non reentrant for the same <code>SoConId</code> .	
<b>Parameters (in):</b>	SoConId	Index of any socket connection that is part of the <code>SoAdSocketConnectionGroup</code> .
	RemoteAddrPtr	Pointer to the structure containing the requested remote IP address and port.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	AssignedSoConIdPtr	Pointer to the <code>SoAd_SoConIdType</code> where the index of the socket connection configured with the remote address ( <code>RemoteAddrPtr</code> ) shall be stored.
<b>Return value:</b>	Std_ReturnType	<code>E_OK</code> : The request was accepted. <code>E_NOT_OK</code> : The request was rejected, <code>AssignedSoConIdPtr</code> remains unchanged.
<b>Description:</b>	This API service shall either return the socket connection index of the <code>SoAdSocketConnectionGroup</code> where the specified remote address (IP address and port) is set or assign the remote address to an unused socket connection from the same <code>SoAdSocketConnectionGroup</code> .	

] ()

[SWS\_SoAd\_00672] If development error detection is enabled: `SoAd_SetUniqueRemoteAddr()` shall check that the service `SoAd_Init()` was previously called. If the check fails, `SoAd_SetUniqueRemoteAddr()` shall raise the development error `SOAD_E_NOTINIT.`] ()

[SWS\_SoAd\_00673] If development error detection is enabled: `SoAd_SetUniqueRemoteAddr()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_SetUniqueRemoteAddr()` shall raise the development error `SOAD_E_INV_ARG.`] ()

[SWS\_SoAd\_00675] The function `SoAd_SetUniqueRemoteAddr()` shall check if one of the socket connections of the socket connection group, identified by `SoConId`, is already configured with the address specified by `RemoteAddrPtr`. In this case, it shall return the socket connection index via `AssignedSoConIdPtr` and return `E_OK.`] ()

[SWS\_SoAd\_00676] If no socket connection is already configured with the address specified by `RemoteAddrPtr`, `SoAd_SetUniqueRemoteAddr()` shall:

- (1) choose an unused socket connection using the best match algorithm described in [SWS\_SoAd\_00680]
- (2) set it to the remote address specified by `RemoteAddrPtr`
- (3) set `AssignedSoConIdPtr` to the index of the chosen socket connection and
- (4) return `E_OK.`

A socket connection is “unused” if its actual remote address has an IP address wildcard and/or port wildcard.] ()

[SWS\_SoAd\_00678] `SoAd_SetUniqueRemoteAddr()` shall reject the request and return `E_NOT_OK` if there are no unused socket connections within the socket connection group identified by `SoConId.`] ()

[SWS\_SoAd\_00700] `SoAd_SetUniqueRemoteAddr()` shall refuse the request if the domain set in `RemoteAddrPtr` does not match the `Tcplp_DomainType` of the local address related to the socket connection identified by `SoConId` and return `E_NOT_OK.` If development error detection is enabled, the service `SoAd_SetUniqueRemoteAddr()` shall also raise the development error `SOAD_E_INV_ARG.`] ()

#### 8.3.4.16 **SoAd\_ReleaseRemoteAddr**

[SWS\_SoAd\_00733] [

<b>Service name:</b>	SoAd_ReleaseRemoteAddr
<b>Syntax:</b>	void SoAd_ReleaseRemoteAddr( SoAd_SoConIdType SoConId )
<b>Service ID[hex]:</b>	0x23
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.
<b>Parameters (in):</b>	SoConId      Index of the socket connection for which the remote address shall be released.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	By this API service the remote address (IP address and port) of the specified socket connection shall be released, i.e. set back to the configured remote address setting.

] ()

[SWS\_SoAd\_00744] [ If development error detection is enabled and SoConId refers to a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to TRUE, the function SoAd\_ReleaseRemoteAddr() shall raise the development error SOAD\_E\_INV\_ARG.] (SRS\_BSW\_00337)

[SWS\_SoAd\_00745] [ The function SoAd\_ReleaseRemoteAddr() shall only immediately proceed if the SoConId refers to a socket connection which is not currently locked by the SoAd. If it is locked, the request shall be postponed to the SoAd\_MainFunction() and executed once the lock is released.] (SRS\_Eth\_00085)

[SWS\_SoAd\_00746] [ The function SoAd\_ReleaseRemoteAddr() shall reset the remote address of the socket connection referred by parameter SoConId to the configured remote address setting.] (SRS\_Eth\_00085)

Note: The intention is to roll back to a wildcard configuration after it was set via SoAd\_UniqueRemoteAddr().

### 8.3.4.17 SoAd\_TpChangeParameter

[SWS\_SoAd\_00508] [

<b>Service name:</b>	SoAd_TpChangeParameter	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_TpChangeParameter(     PduIdType id,     TParameterType parameter,     uint16 value )</pre>	
<b>Service ID[hex]:</b>	0x4b	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	id	Identification of the PDU which the parameter change shall affect.
	parameter	ID of the parameter that shall be changed.
	value	The new value of the parameter.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: The parameter was changed successfully.
		E_NOT_OK: The parameter change was rejected.
<b>Description:</b>	Request to change a specific transport protocol parameter (e.g. block size).	

] ()

[SWS\_SoAd\_00730] [ SoAd\_TpChangeParameter() shall always reject requests by returning E\_NOT\_OK.] ()

### 8.3.4.18 SoAd\_ReadDhcpHostNameOption

#### [SWS\_SoAd\_00681]

<b>Service name:</b>	SoAd_ReadDhcpHostNameOption	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_ReadDhcpHostNameOption(     SoAd_SoConIdType SoConId,     uint8* length,     uint8* data )</pre>	
<b>Service ID[hex]:</b>	0x1A	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection for which the hostname shall be read
<b>Parameters (inout):</b>	length	As input parameter, contains the length of the provided data buffer. Will be overwritten with the length of the actual data.
<b>Parameters (out):</b>	data	Pointer to provided memory buffer the hostname, i.e. the Fully Qualified Domain Name (FQDN) according to IETF RFC 4702/IETF RFC 4704 will be copied to.
<b>Return value:</b>	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
<b>Description:</b>	By this API service an upper layer of the SoAd can read the currently configured hostname, i.e. FQDN option in the DHCP submodule of the TCP/IP stack.	

] ()

[SWS\_SoAd\_00701] If development error detection is enabled:

SoAd\_ReadDhcpHostNameOption() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_ReadDhcpHostNameOption() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00702] If development error detection is enabled:

SoAd\_ReadDhcpHostNameOption() shall check parameter SoConId for being valid. If the check fails, SoAd\_ReadDhcpHostNameOption() shall raise the development error SOAD\_E\_INV\_ARG.] ()

[SWS\_SoAd\_00703] The service SoAd\_ReadDhcpHostNameOption() shall forward the call to TcpIp\_DhcpReadOption() with the parameter Option set to the option code 81 according to IETF RFC 4702, if the socket connection identified by SoConId is related to a local address of the TcpIp\_DomainType TCP\_IP\_AF\_INET.] ()

[SWS\_SoAd\_00704] The service SoAd\_ReadDhcpHostNameOption() shall forward the call to TcpIp\_DhcpV6ReadOption() with the parameter Option set to the option code 39 according to IETF RFC 4704, if the socket connection identified by SoConId is related to a local address of the TcpIp\_DomainType TCP\_IP\_AF\_INET6.] ()

### 8.3.4.19 SoAd\_WriteDhcpHostNameOption

#### [SWS\_SoAd\_00679] [

<b>Service name:</b>	SoAd_WriteDhcpHostNameOption	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_WriteDhcpHostNameOption(     SoAd_SoConIdType SoConId,     uint8 length,     const uint8* data )</pre>	
<b>Service ID[hex]:</b>	0x1B	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection for which the hostname shall be changed
	length	Length of hostname to be set.
	data	Pointer to memory containing the hostname, i.e. the Fully Qualified Domain Name (FQDN) according to IETF RFC 4702/IETF RFC 4704.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
<b>Description:</b>	By this API service an upper layer of the SoAd can set the hostname, i.e. FQDN option in the DHCP submodule of the TCP/IP stack.	

] ()

[SWS\_SoAd\_00705] If development error detection is enabled: SoAd\_WriteDhcpHostNameOption() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_WriteDhcpHostNameOption() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00706] If development error detection is enabled: SoAd\_WriteDhcpHostNameOption() shall check parameter SoConId for being valid. If the check fails, SoAd\_WriteDhcpHostNameOption() shall raise the development error SOAD\_E\_INV\_ARG.] ()

[SWS\_SoAd\_00707] The service SoAd\_WriteDhcpHostNameOption() shall forward the call to TcpIp\_DhcpWriteOption() with the parameter Option set to the option code 81 according to IETF RFC 4702, if the socket connection identified by SoConId is related to a local address of the TcpIp\_DomainType TCPIP\_AF\_INET.] ()

[SWS\_SoAd\_00708] The service SoAd\_WriteDhcpHostNameOption() shall forward the call to TcpIp\_DhcpV6WriteOption() with the parameter Option set to the option code 39 according to IETF RFC 4704, if the socket connection identified by SoConId is related to a local address of the TcpIp\_DomainType TCPIP\_AF\_INET6.] ()



### 8.3.4.20 SoAd\_GetAndResetMeasurementData

#### [SWS\_SoAd\_00010] [

<b>Service name:</b>	SoAd_GetAndResetMeasurementData	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_GetAndResetMeasurementData (     SoAd_MeasurementIdxType MeasurementIdx,     boolean MeasurementResetNeeded,     uint32* MeasurementDataPtr )</pre>	
<b>Service ID[hex]:</b>	0x45	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	MeasurementIdx	Data index of measurement data
	MeasurementResetNeeded	Flag to trigger a reset of the measurement data
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	MeasurementDataPtr	Reference to data buffer, where to copy measurement data
<b>Return value:</b>	Std_ReturnType	E_OK: successful
		E_NOT_OK: failed
<b>Description:</b>	Allows to read and reset detailed measurement data for diagnostic purposes. Get all MeasurementIdx's at once is not supported. SOAD_MEAS_ALL shall only be used to reset all MeasurementIdx's at once. A NULL_PTR shall be provided for MeasurementDataPtr in this case.	

] ()

[SWS\_SoAd\_00757] [ The function `SoAd_GetAndResetMeasurementData()` shall be pre compile time configurable On/Off by the configuration parameter: `SoAdGetAndResetMeasurementDataApi.`] (SRS\_Eth\_00131)

[SWS\_SoAd\_00748] [ `SoAd_GetAndResetMeasurementData()` shall return measurement data for selected measurement index.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00749] [ For measurement index `SOAD_MEAS_DROP_TCP` `SoAd_GetAndResetMeasurementData()` shall return the number of dropped TCP-Port PDUs.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00750] [ For measurement index `SOAD_MEAS_DROP_UDP` `SoAd_GetAndResetMeasurementData()` shall return the number of dropped UDP-Port PDUs.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00751] [ `SoAd_GetAndResetMeasurementData()` shall return `E_NOT_OK` if the requested measurement index is not supported.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00752] [ `SoAd_GetAndResetMeasurementData()` shall additionally reset the measurement data to 0 if the `MeasurementResetNeeded` is true. The reset shall be applied after measurement data has been read.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00753] [ SoAd\_GetAndResetMeasurementData() shall reset all existing measurement data to 0, if MeasurementResetNeeded is true and measurement index is set to SOAD\_MEAS\_ALL.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00754] [ All measurement data which counts data shall not overrun.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00755] [ SoAd\_GetAndResetMeasurementData() shall accept MeasurementDataPtr set to NULL\_PTR. In this case the measurement data shall not be copied.] (SRS\_Eth\_00131)

[SWS\_SoAd\_00756] [ If development error detection is enabled: SoAd\_GetAndResetMeasurementData() shall check that the service SoAd\_Init() was previously called. If the check fails, SoAd\_GetAndResetMeasurementData() shall raise the development error SOAD\_E\_NOTINIT.] (SRS\_Eth\_00131)

## 8.4 Call-back notifications

In AUTOSAR, the functions a module provides to layers which are placed below the module in the AUTOSAR software layer model, are called 'call-back functions'. Generally, a software entity A (SoAd), which, in order to be informed about some event C in software entity B (TCP/IP stack), is registered as interested in event C at software entity B by calling a register mechanism B provides, and is called by entity B if event C occurs. In AUTOSAR the Call-back is usually implicitly registered by configuration.

The following services of the SoAd are called by the TCP/IP Stack.

### 8.4.1 SoAd\_RxIndication

[SWS\_SoAd\_00097] [

<b>Service name:</b>	SoAd_RxIndication	
<b>Syntax:</b>	<pre>void SoAd_RxIndication(     TcpIp_SocketIdType SocketId,     const TcpIp_SockAddrType* RemoteAddrPtr,     uint8* BufPtr,     uint16 Length )</pre>	
<b>Service ID[hex]:</b>	0x12	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SocketIds. Non reentrant for the same SocketId.	
<b>Parameters (in):</b>	SocketId	Socket identifier of the related local socket resource.
	RemoteAddrPtr	Pointer to memory containing IP address and port of the remote host which sent the data.
	BufPtr	Pointer to the received data.
	Length	Data length of the received TCP segment or UDP datagram.

<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	The TCP/IP stack calls this primitive after the reception of data on a socket. The socket identifier along with configuration information determines which module is to be called.

] ()

[SWS\_SoAd\_00264] [

If development error detection is enabled: SoAd\_RxIndication() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_RxIndication() shall raise the development error SOAD\_E\_NOTINIT. ] ()

[SWS\_SoAd\_00267] [

If development error detection is enabled: SoAd\_RxIndication() shall check parameter SocketId for being valid. If the check fails, SoAd\_RxIndication() shall raise the development error SOAD\_E\_INV\_SOCKETID. ] ()

[SWS\_SoAd\_00268] [

If development error detection is enabled: SoAd\_RxIndication() shall check parameter RemoteAddrPtr for being valid. If the check fails, SoAd\_RxIndication() shall raise the development error SOAD\_E\_INV\_ARG. ] ()

## 8.4.2 SoAd\_CopyTxData

[SWS\_SoAd\_00523] [

<b>Service name:</b>	SoAd_CopyTxData	
<b>Syntax:</b>	<pre>BufReq_ReturnType SoAd_CopyTxData(     TcpIp_SocketIdType SocketId,     uint8* BufPtr,     uint16 BufLength )</pre>	
<b>Service ID[hex]:</b>	0x13	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SocketIds. Non reentrant for the same SocketId.	
<b>Parameters (in):</b>	SocketId	Socket identifier of the related local socket resource.
	BufPtr	Pointer to buffer for transmission data.
	BufLength	Length of provided data buffer.

<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_NOT_OK: Data has not been copied. Request failed. (No further action for TcpIp required. Later the upper layer might either close the socket or retry the transmit request)
<b>Description:</b>	This service requests to copy data for transmission to the buffer indicated. This call is triggered by TcpIp_Transmit(). Note: The call to <Up>_CopyTxData() may happen in the context of TcpIp_Transmit().	

] ()

[SWS\_SoAd\_00632] [ If development error detection is enabled: SoAd\_CopyTxData() shall check that the service SoAd\_Init was previously called. If the check fails, SoAd\_CopyTxData() shall raise the development error SOAD\_E\_NOTINIT.] ()

[SWS\_SoAd\_00633] [ If development error detection is enabled: SoAd\_CopyTxData() shall check parameter SocketId for being valid. If the check fails, SoAd\_CopyTxData() shall raise the development error SOAD\_E\_INV\_SOCKETID. ] ()

### 8.4.3 SoAd\_TxConfirmation

[SWS\_SoAd\_00098] [

<b>Service name:</b>	SoAd_TxConfirmation	
<b>Syntax:</b>	<pre>void SoAd_TxConfirmation(     TcpIp_SocketIdType SocketId,     uint16 Length )</pre>	
<b>Service ID[hex]:</b>	0x14	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SocketIds. Non reentrant for the same SocketId.	
<b>Parameters (in):</b>	SocketId	Socket identifier of the related local socket resource.
	Length	Number of transmitted data bytes.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	The TCP/IP stack calls this function after the data has been acknowledged by the peer for TCP.  Caveats: The upper layer might not be able to determine exactly which data bytes have been confirmed.	

] ()

[SWS\_SoAd\_00269] [

If development error detection is enabled: SoAd\_TxConfirmation() shall check that the service SoAd\_Init was previously called. If the check fails,

`SoAd_TxConfirmation()` shall raise the development error `SOAD_E_NOTINIT.` ]  
( )

**[SWS\_SoAd\_00270]** [

If development error detection is enabled: `SoAd_TxConfirmation()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TxConfirmation()` shall raise the development error `SOAD_E_INV_SOCKETID.` ] ( )

**[SWS\_SoAd\_00271]** [

If development error detection is enabled: `SoAd_TxConfirmation()` shall check parameter `Length` for being valid. If the check fails, `SoAd_TxConfirmation()` shall raise the development error `SOAD_E_INV_ARG.` ] ( )

## 8.4.4 SoAd\_TcpAccepted

**[SWS\_SoAd\_00099]** [

<b>Service name:</b>	SoAd_TcpAccepted	
<b>Syntax:</b>	<pre>Std_ReturnType SoAd_TcpAccepted(     TcpIp_SocketIdType SocketId,     TcpIp_SocketIdType SocketIdConnected,     const TcpIp_SockAddrType* RemoteAddrPtr )</pre>	
<b>Service ID[hex]:</b>	0x15	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	SocketId	Socket identifier of the related local socket resource which has been used at <code>Tcplp_Bind()</code>
	SocketIdConnected	Socket identifier of the local socket resource used for the established connection.
	RemoteAddrPtr	IP address and port of the remote host.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	Result of operation <code>E_OK</code> upper layer accepts the established connection <code>E_NOT_OK</code> upper layer refuses the established connection, <code>Tcplp</code> stack shall close the connection.
<b>Description:</b>	This service gets called if the stack put a socket into the listen mode before (as server) and a peer connected to it (as client). In detail: The TCP/IP stack calls this function after a socket was set into the listen state with <code>Tcplp_TcpListen()</code> and a TCP connection is requested by the peer.	

] ( )

**[SWS\_SoAd\_00272]** [

If development error detection is enabled: `SoAd_TcpAccepted()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TcpAccepted()` shall raise the development error `SOAD_E_NOTINIT.` ] ( )

**[SWS\_SoAd\_00273]** [

If development error detection is enabled: `SoAd_TcpAccepted()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TcpAccepted()` shall raise the development error `SOAD_E_INV_SOCKETID`. ] ( )

#### 8.4.5 SoAd\_TcpConnected

##### [SWS\_SoAd\_00100] [

<b>Service name:</b>	SoAd_TcpConnected	
<b>Syntax:</b>	<pre>void SoAd_TcpConnected(     TcpIp_SocketIdType SocketId )</pre>	
<b>Service ID[hex]:</b>	0x16	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	SocketId	Socket identifier of the related local socket resource.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	<p>This service gets called if the stack initiated a TCP connection before (as client) and the peer (the server) acknowledged the connection set up. In detail:</p> <p>The TCP/IP stack calls this function after a socket was requested to connect with <code>TcpIp_TcpConnect()</code> and a TCP connection is confirmed by the peer. The parameter value of <code>SocketId</code> equals the <code>SocketId</code> value of the preceding <code>TcpIp_TcpConnect()</code> call.</p>	

] ( )

##### [SWS\_SoAd\_00274] [

If development error detection is enabled: `SoAd_TcpConnected()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TcpConnected()` shall raise the development error `SOAD_E_NOTINIT`. ] ( )

##### [SWS\_SoAd\_00275] [

If development error detection is enabled: `SoAd_TcpConnected()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TcpConnected()` shall raise the development error `SOAD_E_INV_SOCKETID`. ] ( )

#### 8.4.6 SoAd\_TcplpEvent

##### [SWS\_SoAd\_00146] [

<b>Service name:</b>	SoAd_TcplpEvent	
<b>Syntax:</b>	<pre>void SoAd_TcpIpEvent(     TcpIp_SocketIdType SocketId,     TcpIp_EventType Event )</pre>	
<b>Service ID[hex]:</b>	0x17	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	SocketId	Socket identifier of the related local socket resource.
	Event	This parameter contains a description of the event just encountered.

<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This service gets called if the stack encounters a condition described by the values in Event.

] () [SWS\_SoAd\_00276] [

If development error detection is enabled: `SoAd_TcpIpEvent()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TcpIpEvent()` shall raise the development error `SOAD_E_NOTINIT`. ] ()

[SWS\_SoAd\_00277] [

If development error detection is enabled: `SoAd_TcpIpEvent()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TcpIpEvent()` shall raise the development error `SOAD_E_INV_SOCKETID`. ] ()

[SWS\_SoAd\_00278] [

If development error detection is enabled: `SoAd_TcpIpEvent()` shall check parameter `Event` for being valid. If the check fails, `SoAd_TcpIpEvent()` shall raise the development error `SOAD_E_INV_ARG`. ] ()

## 8.4.7 SoAd\_LocalIpAddrAssignmentChg

[SWS\_SoAd\_00209] [

<b>Service name:</b>	SoAd_LocalIpAddrAssignmentChg	
<b>Syntax:</b>	<pre>void SoAd_LocalIpAddrAssignmentChg(     TcpIp_LocalAddrIdType IpAddrId,     TcpIp_IpAddrStateType State )</pre>	
<b>Service ID[hex]:</b>	0x18	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	IpAddrId	IP address Identifier, representing an IP address specified in the TcpIp module configuraiton (e.g. static IPv4 address on EthIf controller 0).
	State	state of IP address assignment
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This service gets called by the TCP/IP stack if an IP address assignment changes (i.e. new address assigned or assigned address becomes invalid).	

] () [SWS\_SoAd\_00279] [

If development error detection is enabled: `SoAd_LocalIpAddrAssignmentChg()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_LocalIpAddrAssignmentChg()` shall raise the development error `SOAD_E_NOTINIT`. ] ()



[SWS\_SoAd\_00729] If `SoAd_LocalIpAddressAssignmentChg()` is called with the parameter `IpAddressId` set to a local address which is not referenced by any `SoAdSocketConnectionGroup`, `SoAd` shall ignore the notification and return without any further action. ] ( )

## 8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

### 8.5.1 Terms and definitions

For details refer to the chapter 8.5 “Scheduled functions” in *SWS\_BSWGeneral*.

### 8.5.2 SoAd\_MainFunction

[SWS\_SoAd\_00121] [

<b>Service name:</b>	SoAd_MainFunction
<b>Syntax:</b>	void SoAd_MainFunction( void )
<b>Service ID[hex]:</b>	0x19
<b>Description:</b>	Schedules the Socket Adaptor. (Entry point for scheduling)

] ( ) [SWS\_SoAd\_00131] [

The main function for scheduling the `SoAd` (Entry point for scheduling) shall be called by the Schedule Manager according to the configured call period. ] ( )

[SWS\_SoAd\_00176] [

The call period of the `SoAd_MainFunction()` shall be determined by configuration parameter `SOAD_MAINFUNCTION_PERIOD`. ] ( )

[SWS\_SoAd\_00283] [

If development error detection is enabled: `SoAd_MainFunction()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_MainFunction()` shall raise the development error `SOAD_E_NOTINIT`. ] ( )

## 8.6 Expected Interfaces

In this chapter all interfaces required by the `SoAd` from other modules are listed.

### 8.6.1 Mandatory Interfaces

This chapter defines all interfaces which are required by the `SoAd` to fulfill the core functionality of the `SoAd` module.

[SWS\_SoAd\_00504] [

API function	Description
Tcplp_Bind	By this API service the TCP/IP stack is requested to bind a UDP or TCP socket to a local resource.
Tcplp_ChangeParameter	By this API service the TCP/IP stack is requested to change a parameter of a socket. E.g. the Nagle algorithm may be controlled by this API.
Tcplp_Close	By this API service the TCP/IP stack is requested to close the socket and release all related resources.
Tcplp_GetCtrlIdx	Tcplp_GetCtrlIdx returns the index of the controller related to LocalAddrId.
Tcplp_GetIpAddr	Obtains the local IP address actually used by LocalAddrId, the netmask and default router
Tcplp_GetPhysAddr	Obtains the physical source address used by the EthIf controller implicitly specified via LocalAddrId.
Tcplp_GetRemotePhysAddr	Tcplp_GetRemotePhysAddr queries the IP/physical address translation table specified by CtrlIdx and returns the physical address related to the IP address specified by IpAddrPtr. In case no physical address can be retrieved and parameter initRes is TRUE, address resolution for the specified IP address is initiated on the local network.
Tcplp_ReleaseIpAddrAssignment	By this API service the local IP address assignment for the IP address specified by LocalAddrId shall be released.
Tcplp_RequestComMode	By this API service the TCP/IP stack is requested to change the Tcplp state of the communication network identified by EthIf controller index.
Tcplp_RequestIpAddrAssignment	By this API service the local IP address assignment for the IP address specified by LocalAddrId shall be initiated.
Tcplp_SoAdGetSocket	By this API service the TCP/IP stack is requested to allocate a new socket. Note: Each accepted incoming TCP connection also allocates a socket resource.
Tcplp_TcpConnect	By this API service the TCP/IP stack is requested to establish a TCP connection to the configured peer.
Tcplp_TcpListen	By this API service the TCP/IP stack is requested to listen on the TCP socket specified by the socket identifier.
Tcplp_TcpReceived	By this API service the reception of socket data is confirmed to the TCP/IP stack.
Tcplp_TcpTransmit	This service requests transmission of data via TCP to a remote node. The transmission of the data is decoupled.  Note: The TCP segment(s) are sent dependent on runtime factors (e.g. receive window) and configuration parameter (e.g. Nagle algorithm) .
Tcplp_UdpTransmit	This service transmits data via UDP to a remote node. The transmission of the data is immediately performed with this function call by forwarding it to EthIf.

] ()

## 8.6.2 Optional Interfaces

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

[SWS\_SoAd\_00692]

API function	Description
Det_ReportError	Service to report development errors.
PduR_SoAdTpCopyRxData	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.
PduR_SoAdTpCopyTxData	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
Tcplp_DhcpReadOption	By this API service the TCP/IP stack retrieves DHCP option data identified by parameter option for already received DHCP options.
Tcplp_DhcpV6ReadOption	By this API service the TCP/IP stack retrieves DHCPv6 option data identified by parameter option for already received DHCPv6 options.
Tcplp_DhcpV6WriteOption	By this API service the TCP/IP stack writes the DHCPv6 option data identified by parameter option.
Tcplp_DhcpWriteOption	By this API service the TCP/IP stack writes the DHCP option data identified by parameter option.

] ()

## 8.6.3 Configurable interfaces

In this chapter all interfaces are listed where the target function could be configured. The target function is a call-back function implemented by an upper layer module. The function names contain a tag <Up> that is replaced with the module name abbreviation of the concrete upper layer module and two configurable infix [SoAd] and [If] or [Tp].

[SWS\_SoAd\_00538] For each configurable interface SoAd shall determine the function name by replacing the tag <Up> with the module name abbreviation of the related upper layer module (as specified in the SoAdBSWModules container using the SoAdBswModuleRef reference parameter) and using the two infix according to the configuration parameters SoAdUseCallerInfix and SoAdUseTypeInfix.] ()

The ServiceID of the functions defined in this chapter are specified at the upper layer module implementing the functions.

### 8.6.3.1 <Up>\_[SoAd][If]RxIndication

[SWS\_SoAd\_00106] [

<b>Service name:</b>	<Up>_[SoAd][If]RxIndication	
<b>Syntax:</b>	<pre>void &lt;Up&gt;_[SoAd][If]RxIndication(     PduIdType RxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Indication of a received PDU from a lower layer communication interface module.	

] ()

### 8.6.3.2 <Up>\_[SoAd][If]TriggerTransmit

[SWS\_SoAd\_00663] [

<b>Service name:</b>	<Up>_[SoAd][If]TriggerTransmit	
<b>Syntax:</b>	<pre>Std_ReturnType &lt;Up&gt;_[SoAd][If]TriggerTransmit(     PduIdType TxPduId,     PduInfoType* PduInfoPtr )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLength. On return, the service will indicate the length of the copied SDU data in SduLength.
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes.
		E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
<b>Description:</b>	<p>Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr-&gt;SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr-&gt;SduDataPtr and update the length of the actual copied data in PduInfoPtr-&gt;SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.</p>	

] ()

### 8.6.3.3 <Up>\_[SoAd][If]TxConfirmation

[SWS\_SoAd\_00107] [

<b>Service name:</b>	<Up>_[SoAd][If]TxConfirmation	
<b>Syntax:</b>	<pre>void &lt;Up&gt;_[SoAd][If]TxConfirmation(     PduIdType TxPduId,     Std_ReturnType result )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in):</b>	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	

] ()

#### 8.6.3.4 <Up>\_[SoAd][Tp]StartOfReception

[SWS\_SoAd\_00138] [

<b>Service name:</b>	<Up>_[SoAd][Tp]StartOfReception	
<b>Syntax:</b>	<pre>BufReq_ReturnType &lt;Up&gt;_[SoAd][Tp]StartOfReception(     PduIdType id,     const PduInfoType* info,     PduLengthType TpSduLength,     PduLengthType* bufferSizePtr )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.

<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.
<b>Return value:</b>	BufReq_ReturnType	<p>BUFREQ_OK: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr.</p> <p>BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged.</p> <p>BUFREQ_E_OVFL: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.</p>
<b>Description:</b>	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSduLength equal to 0.	

] ()

### 8.6.3.5 <Up>\_[SoAd][Tp]CopyRxData

[SWS\_SoAd\_00139] [

<b>Service name:</b>	<Up>_[SoAd][Tp]CopyRxData	
<b>Syntax:</b>	<pre>BufReq_ReturnType &lt;Up&gt;_[SoAd][Tp]CopyRxData (     PduIdType id,     const PduInfoType* info,     PduLengthType* bufferSizePtr )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	Identification of the received I-PDU.
	info	<p>Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength).</p> <p>An SduLength of 0 can be used to query the current amount of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.</p>
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	bufferSizePtr	Available receive buffer after data has been copied.
<b>Return value:</b>	BufReq_ReturnType	<p>BUFREQ_OK: Data copied successfully</p> <p>BUFREQ_E_NOT_OK: Data was not copied because an error occurred.</p>
<b>Description:</b>	<p>This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer.</p> <p>Each call to this function provides the next part of the I-PDU data.</p> <p>The size of the remaining data is written to the position indicated by bufferSizePtr.</p>	

] ()

### 8.6.3.6 <Up>\_[SoAd][Tp]RxIndication

[SWS\_SoAd\_00180] [

<b>Service name:</b>	<Up>_[SoAd][Tp]TpRxIndication	
<b>Syntax:</b>	<pre>void &lt;Up&gt;_[SoAd][Tp]TpRxIndication(     PduIdType id,     Std_ReturnType result )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	Identification of the received I-PDU.
	result	Result of the reception.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.	

] ()

### 8.6.3.7 <Up>\_[SoAd][Tp]CopyTxData

[SWS\_SoAd\_00137] [

<b>Service name:</b>	<Up>_[SoAd][Tp]CopyTxData	
<b>Syntax:</b>	<pre>BufReq_ReturnType &lt;Up&gt;_[SoAd][Tp]CopyTxData(     PduIdType id,     const PduInfoType* info,     RetryInfoType* retry,     PduLengthType* availableDataPtr )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	Identification of the transmitted I-PDU.
	info	<p>Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength).</p> <p>If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned. The lower layer module may retry the call.</p> <p>An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.</p>
	retry	<p>This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems.</p> <p>If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element.</p> <p>If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery.</p> <p>TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later.</p> <p>TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In</p>



		this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
<b>Return value:</b>	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
<b>Description:</b>	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	

] ()

### 8.6.3.8 <Up>\_[SoAd][Tp]TxConfirmation

[SWS\_SoAd\_00181] [

<b>Service name:</b>	<Up>_[SoAd][Tp]TxConfirmation	
<b>Syntax:</b>	<pre>void &lt;Up&gt;_[SoAd][Tp]TxConfirmation(     PduIdType id,     Std_ReturnType result )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	id	Identification of the transmitted I-PDU.
	result	Result of the transmission of the I-PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.	

] ()

### 8.6.3.9 <Up>\_SoConModeChg

[SWS\_SoAd\_00514] [

<b>Service name:</b>	<Up>_SoConModeChg	
<b>Syntax:</b>	<pre>void &lt;Up&gt;_SoConModeChg(     SoAd_SoConIdType SoConId,     SoAd_SoConModeType Mode )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection with the mode change.
	Mode	new socket connection mode
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Notification about a SoAd socket connection state change, e.g. socket connection gets online	

] ()

#### 8.6.3.10 <Up>\_LocalIpAddrAssignmentChg

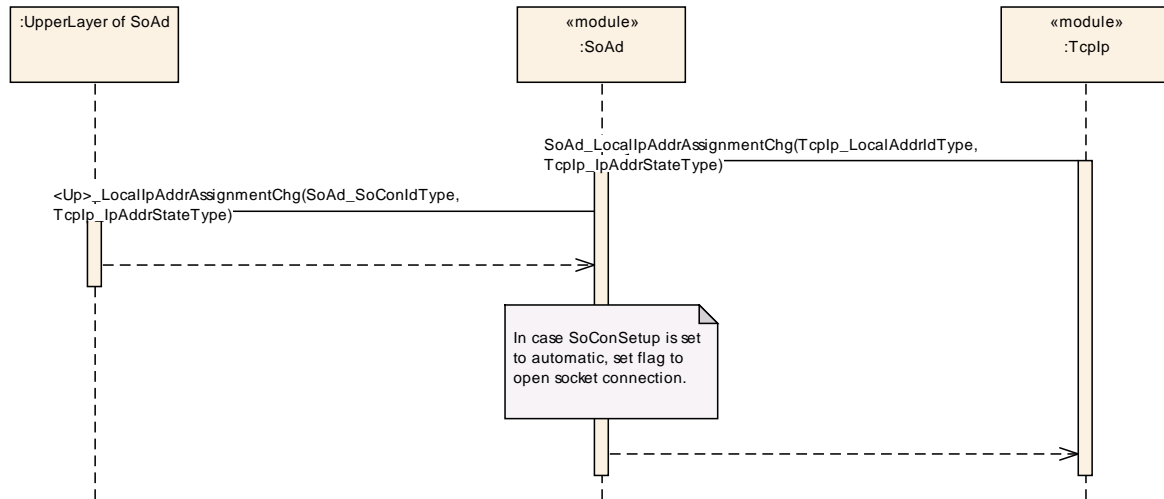
[SWS\_SoAd\_00513] [

<b>Service name:</b>	<Up>_LocalIpAddrAssignmentChg	
<b>Syntax:</b>	<pre>void &lt;Up&gt;_LocalIpAddrAssignmentChg(     SoAd_SoConIdType SoConId,     TcpIp_IpAddrStateType State )</pre>	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in):</b>	SoConId	socket connection index specifying the socket connection where the IP address assignment has changed
	State	state of IP address assignment
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	This function gets called by the SoAd if an IP address assignment related to a socket connection changes (i.e. new address assigned or assigned address becomes invalid).	

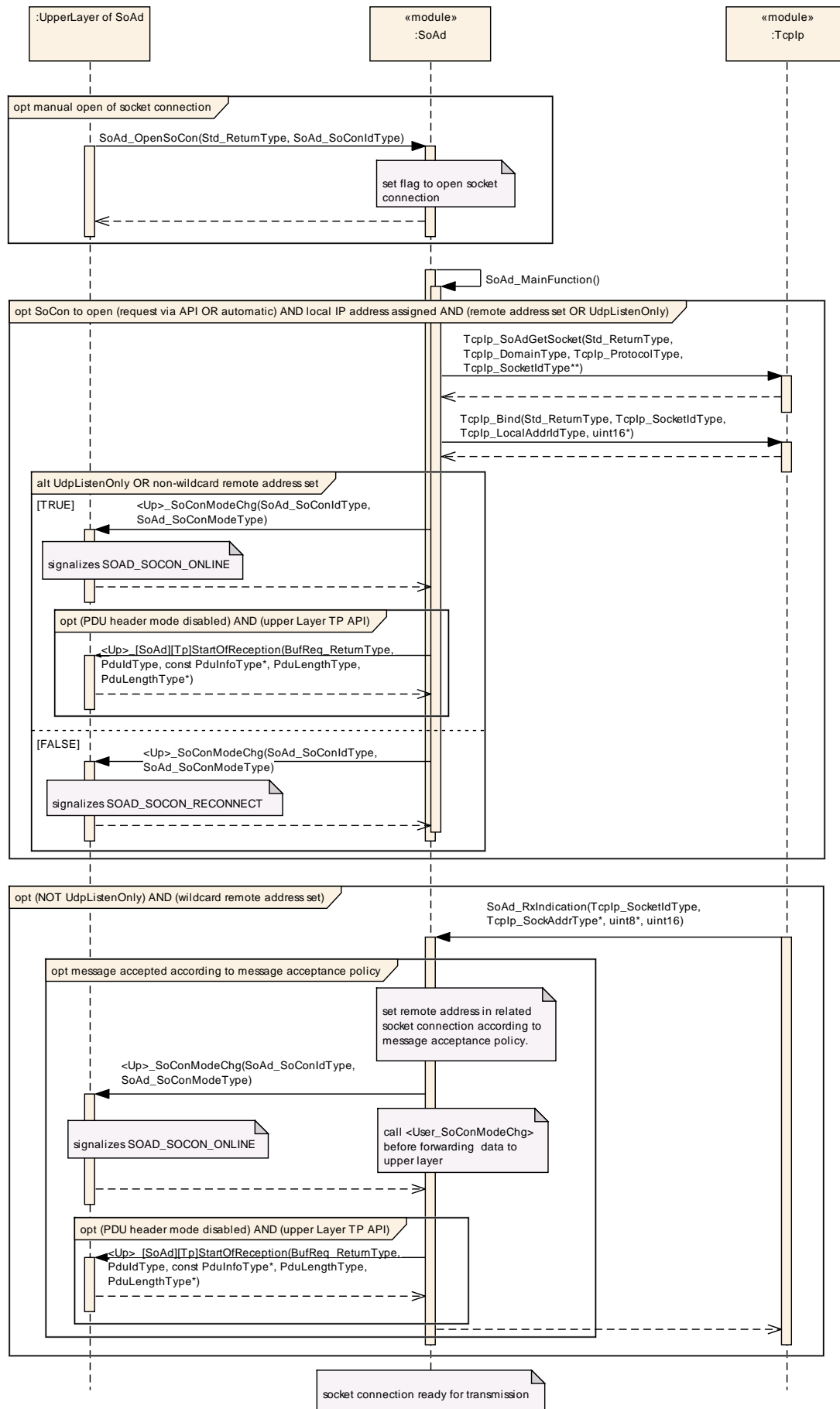
] ()

## 9 Sequence diagrams and Transition Tables

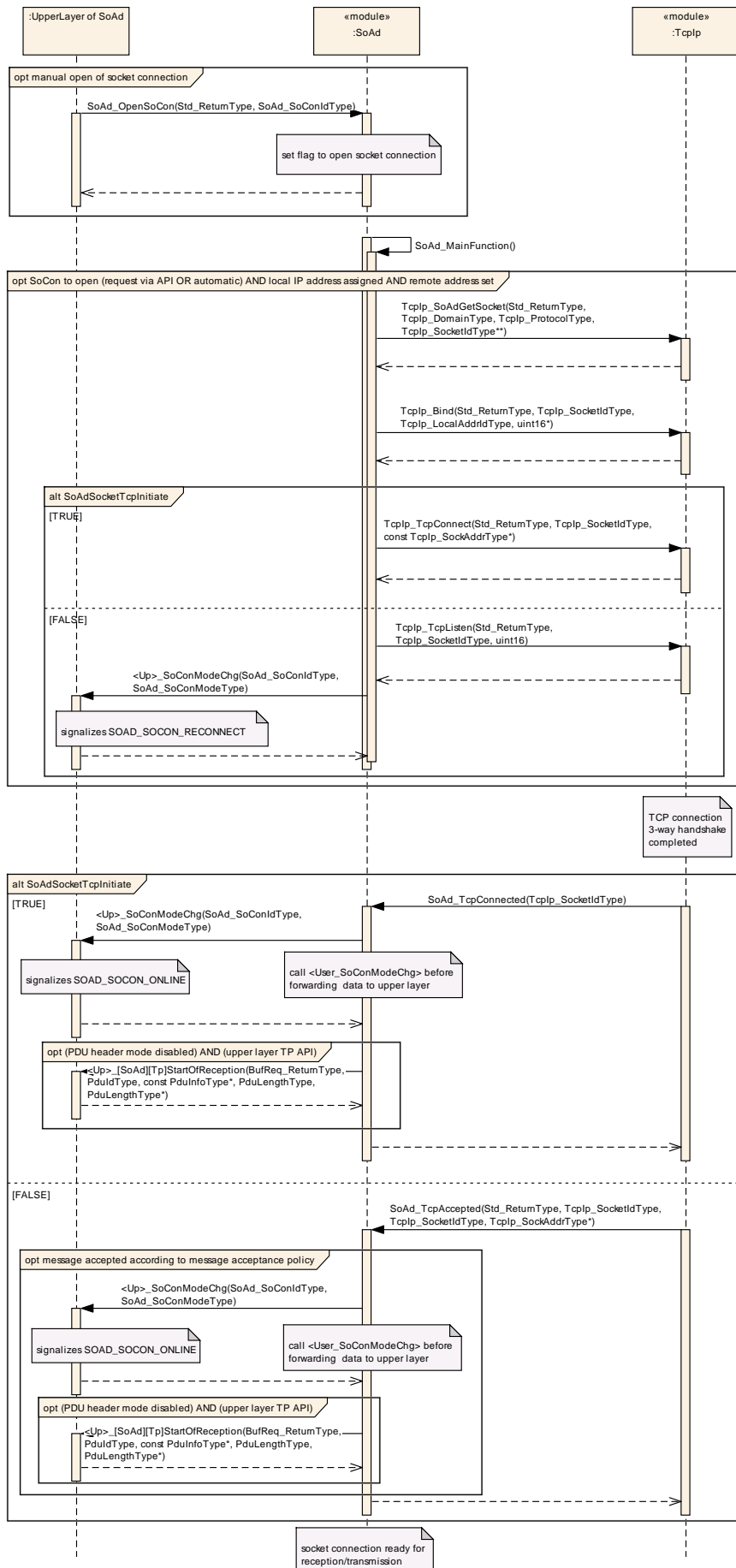
### 9.1 Address – Assignment



## 9.2 Socket Connection Setup – UDP

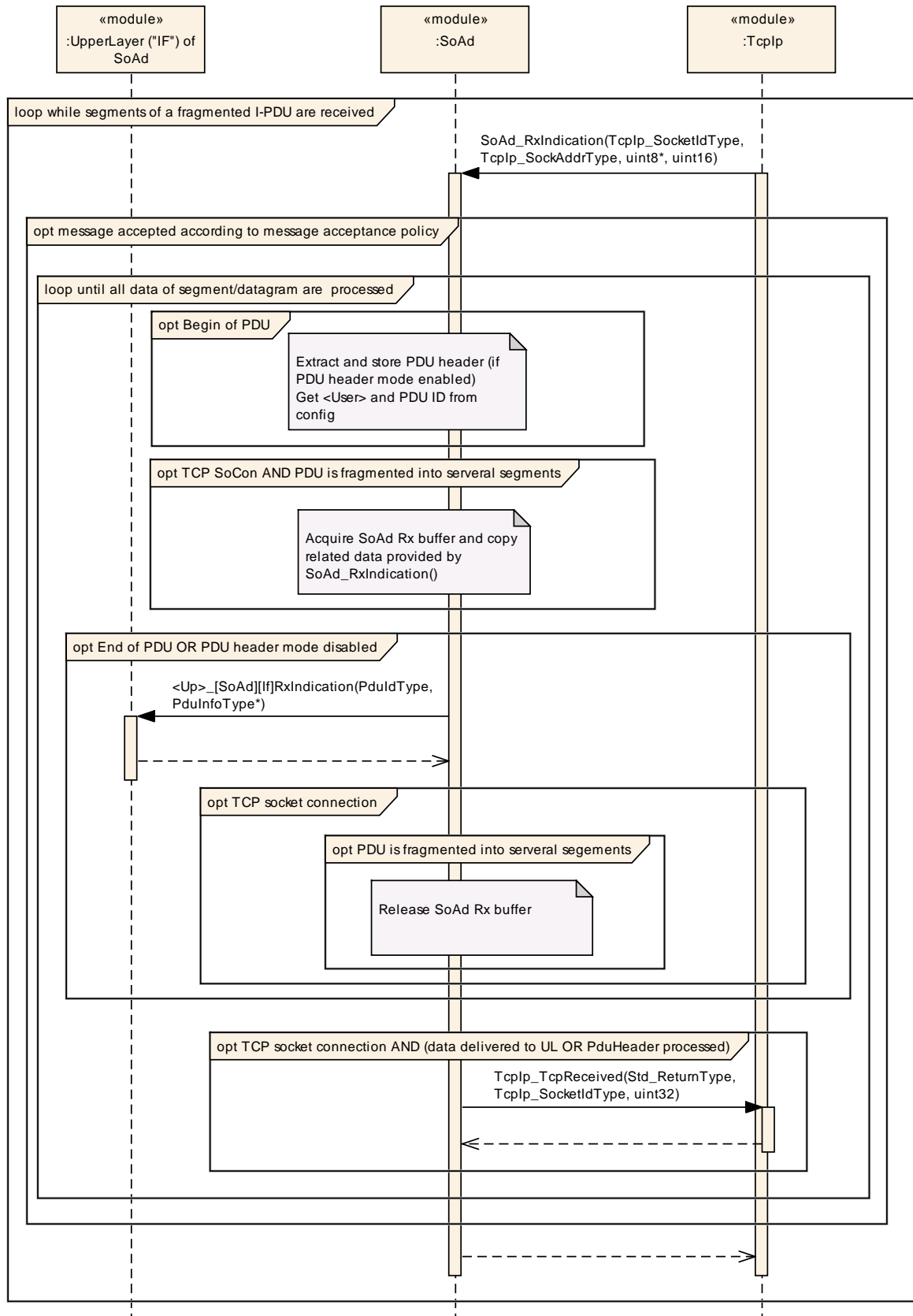


### 9.3 Socket Connection Setup – TCP

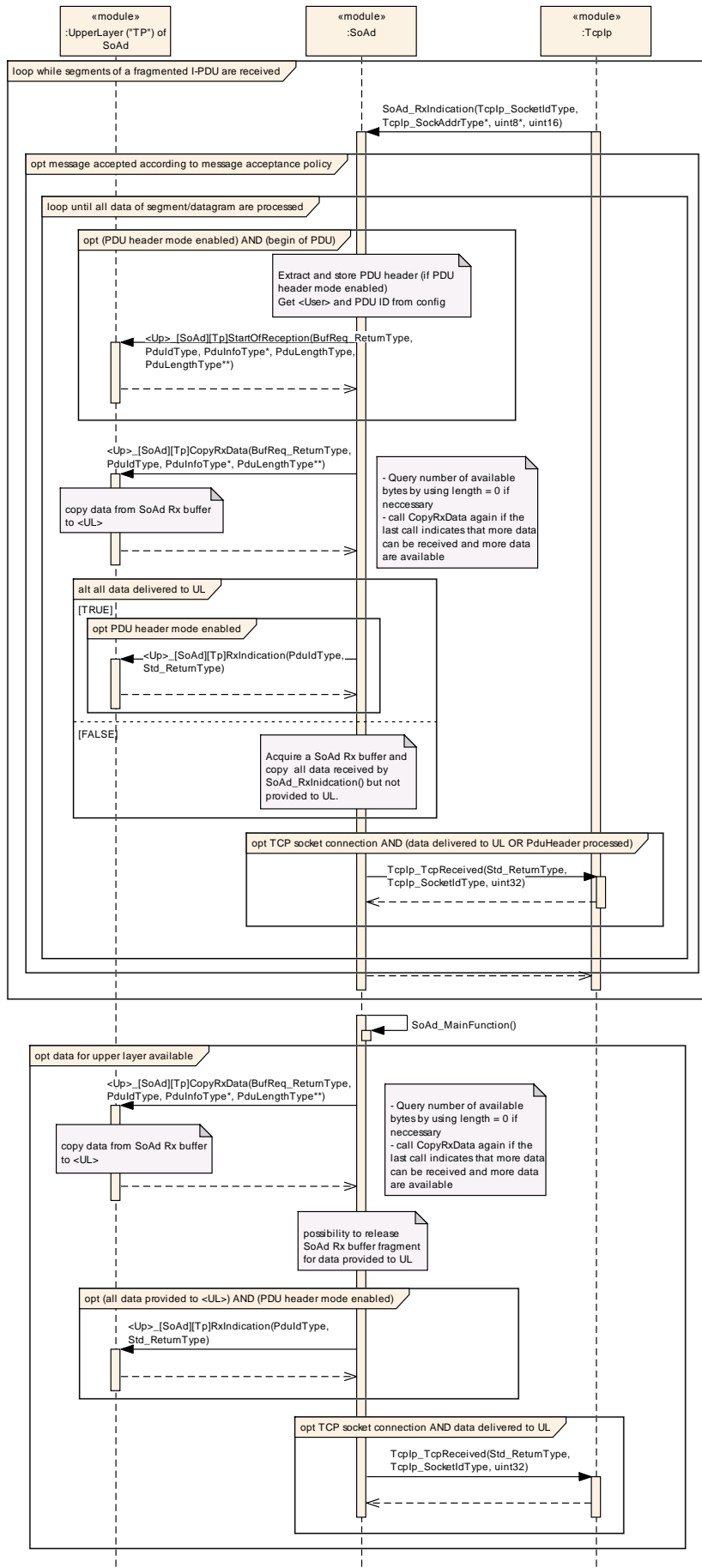




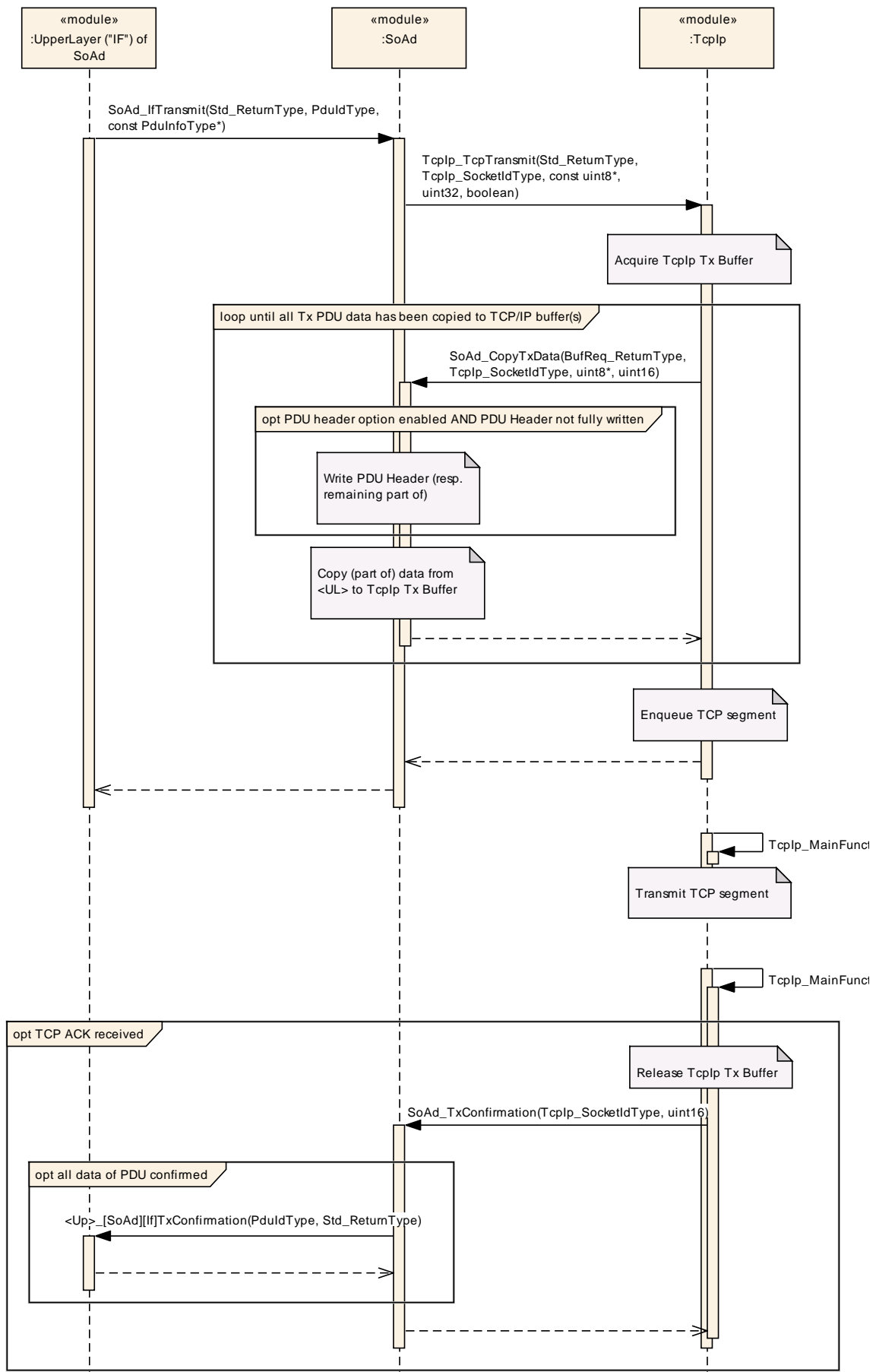
## 9.4 Reception – Upper Layer If API



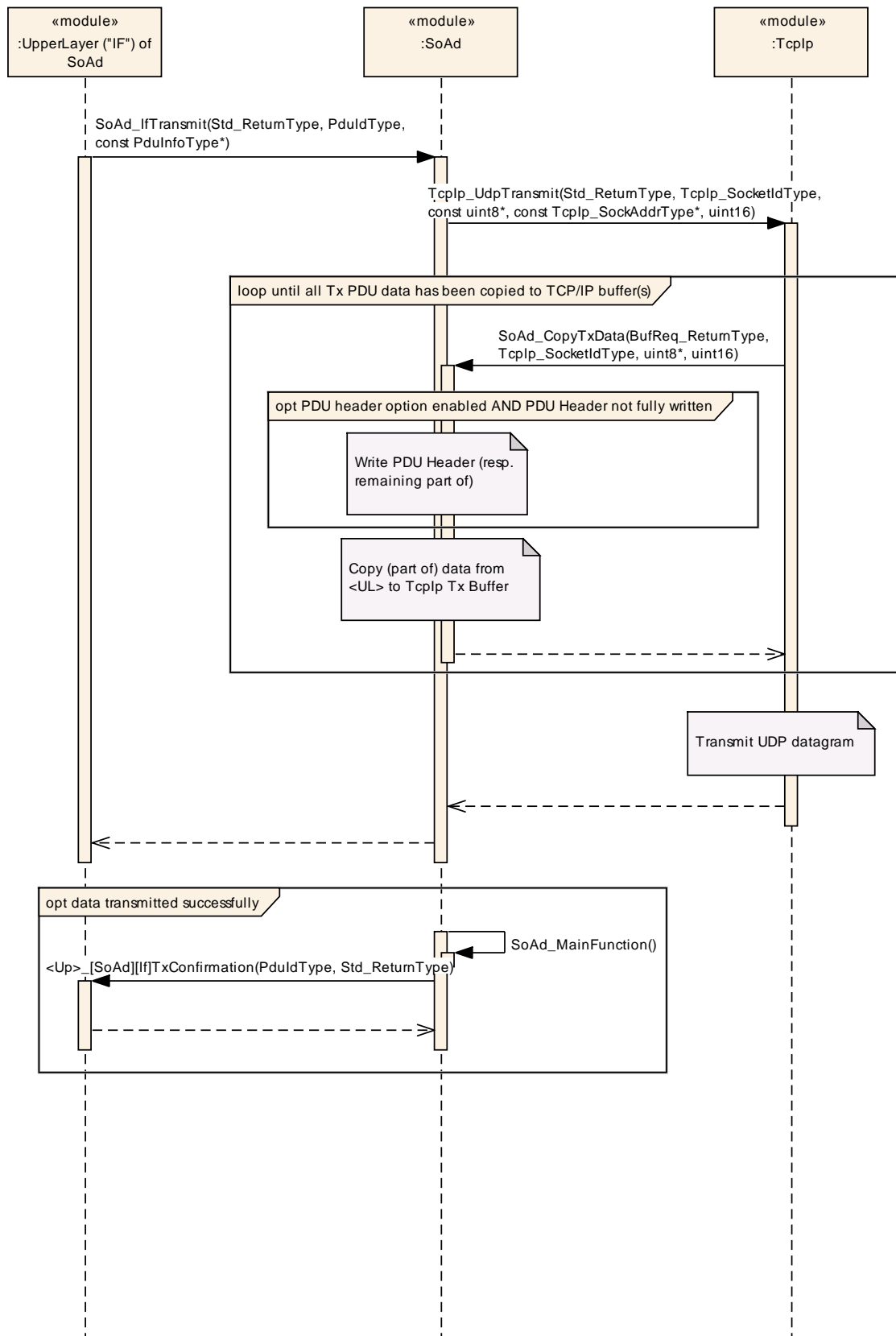
## 9.5 Reception – Upper Layer TP API



## 9.6 Transmission – Upper Layer If API – TCP

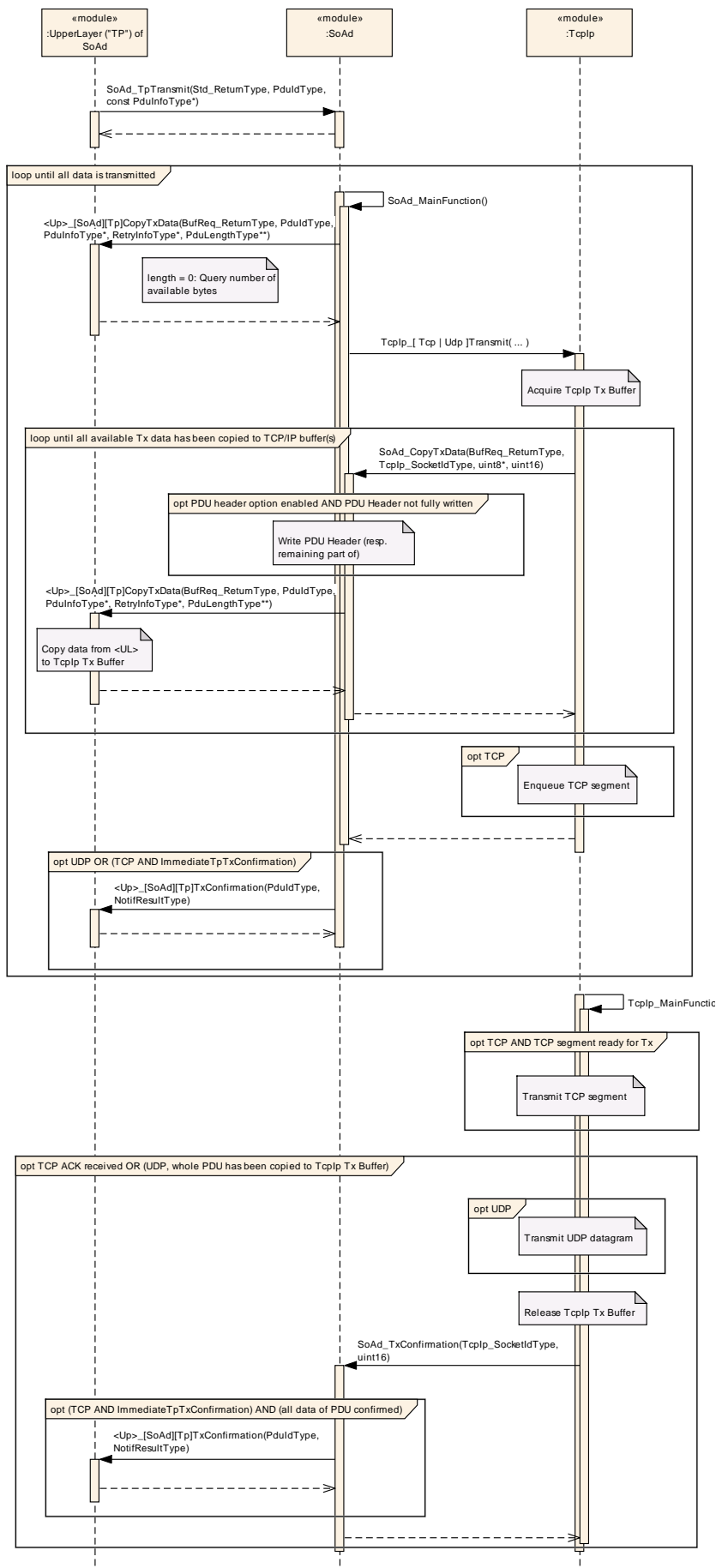


## 9.7 Transmission – Upper Layer If API – UDP



## 9.8 Transmission – Upper Layer TP API





## 10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of module SoAd.

Chapter 10.3 specifies published information of module SoAd.

### 10.1 How to read this chapter

For details refer to the chapter 10.1 "Introduction to configuration specification" in *SWS\_BSWGeneral*.

## 10.2 Containers and configuration parameters

The configuration parameters as defined in this chapter are used to create a data model for an AUTOSAR tool chain. The realization in the code is implementation specific.

The configuration parameters are divided into parameters used to enable features, parameters affecting all instances of the UdpNm and parameters affecting the respective instances of the UdpNm.

[SWS\_SoAd\_00001] [All configuration items shall be located outside the kernel of the module. ] ( )

[SWS\_SoAd\_00208] [All timing parameters given as EcucFloatParamDef in unit seconds in the configuration, shall be converted to integer multiples of the parameter SoAdMainFunctionPeriod.] ( )

### 10.2.1 SoAd

<b>SWS Item</b>	<b>ECUC_SoAd_00001 :</b>
<b>Module Name</b>	SoAd
<b>Module Description</b>	Configuration of the SoAd (Socket Adaptor) module.
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdBswModules	0..*	Each container describes a specific BSW module that the SoAd shall interface to.
SoAdConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR SoAd module.
SoAdGeneral	1	This container contains all global configuration parameters of SoAd.

## 10.2.2 SoAdBswModules

<b>SWS Item</b>	<b>ECUC_SoAd_00102 :</b>
<b>Container Name</b>	SoAdBswModules
<b>Description</b>	Each container describes a specific BSW module that the SoAd shall interface to.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00104 :</b>		
<b>Name</b>	SoAdIf		
<b>Description</b>	Specifies if the BSW module supports the Communication Interface APIs or not. Value true means that the APIs are supported. A module can have both Communication Interface APIs and Transport Protocol APIs (e.g. the PduR module).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00145 :</b>		
<b>Name</b>	SoAdIfTriggerTransmit		
<b>Description</b>	Specifies if the BSW module supports the TriggerTransmit API or not. Value true means that the API is supported.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00106 :</b>		
<b>Name</b>	SoAdIfTxConfirmation		
<b>Description</b>	Specifies if the BSW module supports the TxConfirmation API or not. Value true means that the API is supported.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>SWS Item</b>	<b>ECUC_SoAd_00143 :</b>		
<b>Name</b>	SoAdLocalIpAddrAssignmentChg		
<b>Description</b>	Specifies if the BSW module supports the LocalIpAddrAssignmentChg API or not. Value true means that the API is supported.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00107 :</b>		
<b>Name</b>	SoAdSoConModeChg		
<b>Description</b>	Specifies if the BSW module supports the SoConModeChg API or not. Value true means that the API is supported.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00105 :</b>		
<b>Name</b>	SoAdTp		
<b>Description</b>	Specifies if the BSW module supports the TransportProtocol APIs or not. Value true means that the APIs are supported. A module can have both Communication Interface APIs and Transport Protocol APIs (e.g. the PduR module).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00128 :</b>		
<b>Name</b>	SoAdUseCallerInfix		
<b>Description</b>	Specifies if SoAd shall use (TRUE) the infix "SoAd" when calling an upper layer module function or not (FALSE). E.g. if SoAdUseCallerInfix is TRUE for the upper layer "ABC" then SoAd will call ABC_SoAdIfRxIndication() otherwise SoAd would call ABC_IfRxIndication().		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>SWS Item</b>	<b>ECUC_SoAd_00129 :</b>		
<b>Name</b>	SoAdUseTypeInfix		
<b>Description</b>	Specifies if SoAd shall use (TRUE) the API type infix "Tp" or "If" when calling an upper layer module function or not (FALSE). E.g. if SoAdUseTypeInfix is TRUE for the upper layer "ABC" then SoAd will call ABC_IfRxIndication(), otherwise SoAd would call ABC_RxIndication().		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00124 :</b>		
<b>Name</b>	SoAdBswModuleRef		
<b>Description</b>	<p>This is a reference to one BSW module's configuration (i.e. not the ECUC parameter definition template).  Example, there could be several configurations of PduR and this reference selects one of them.</p> <p>SoAd has to figure out from the structure of the referenced BSW module's configuration, what kind of upper layer he deals with.  In case of a CDD SoAd expects UL-APIs in form of _SoAd&lt;If Tp&gt;&lt;function&gt; and expects CDD Pdu configuration structures according to the Ecu Configuration specification (chapter CDD module\Socket Adaptor).  In case it is one of the standardized AUTOSAR BSW modules, the configuration structures and API names for interaction with SoAd are defined in the corresponding SWS.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	Foreign reference to [ ECUC-MODULE-CONFIGURATION-VALUES ]		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
<b>No Included Containers</b>	

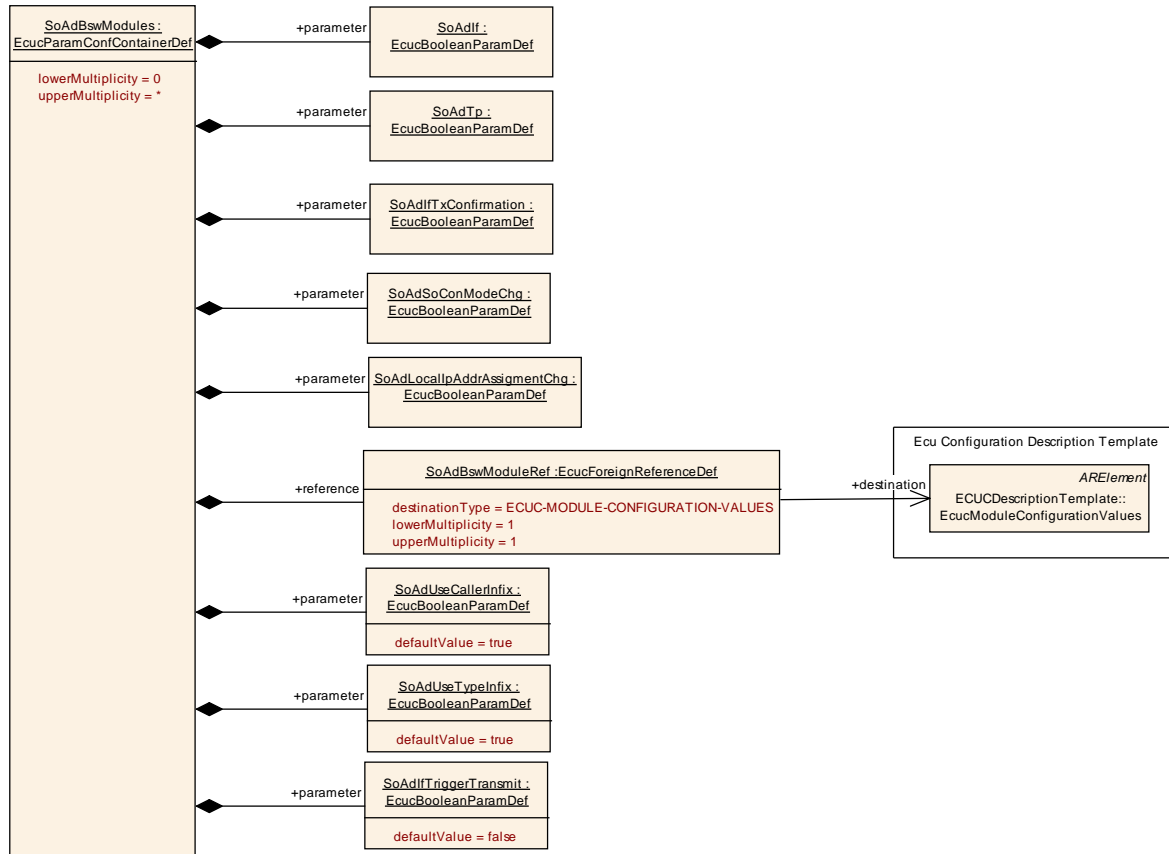


Figure 3: SoAd BswModules container

### 10.2.3 SoAdGeneral

<b>SWS Item</b>	<b>ECUC_SoAd_00003 :</b>
<b>Container Name</b>	SoAdGeneral
<b>Description</b>	This container contains all global configuration parameters of SoAd.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00002 :</b>		
<b>Name</b>	SoAdDevErrorDetect		
<b>Description</b>	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>true: detection and notification is enabled.</li> <li>false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	



	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00162 :</b>		
<b>Name</b>	SoAdGetAndResetMeasurementDataApi		
<b>Description</b>	Enables / Disables the Get and Reset Measurement Data API		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00039 :</b>		
<b>Name</b>	SoAdIPv6AddressEnabled		
<b>Description</b>	Allows for increased memory allocation to store IPv6 addresses. TRUE: Enables support for IPv6 addresses FALSE: Only IPv4 addresses are supported		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00062 :</b>		
<b>Name</b>	SoAdMainFunctionPeriod		
<b>Description</b>	Determines the frequency at which the SoAd_MainFunction() is called in [s].		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	]0 .. INF[		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

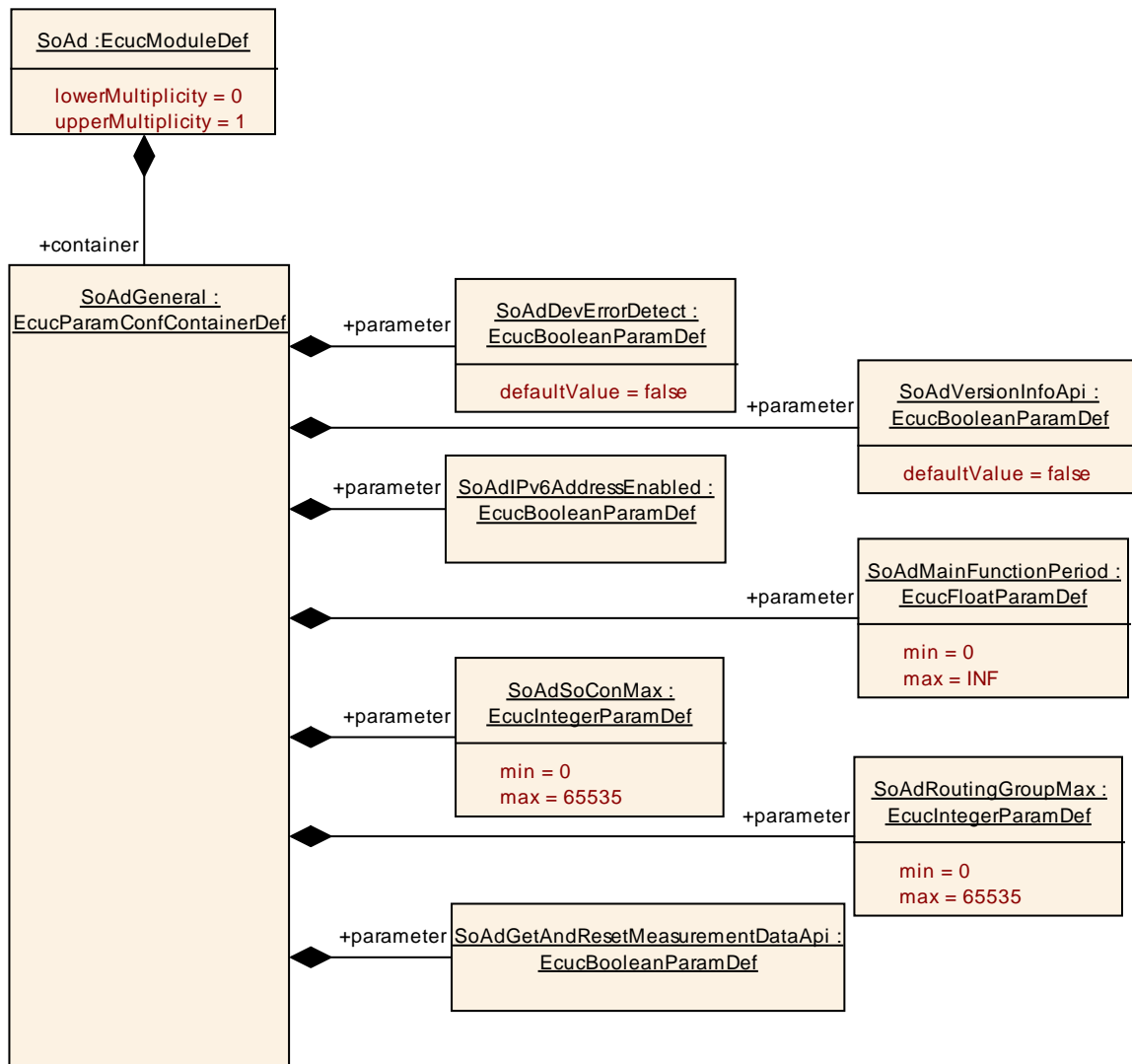
<b>SWS Item</b>	<b>ECUC_SoAd_00127 :</b>		
<b>Name</b>	SoAdRoutingGroupMax		
<b>Description</b>	Specifies the maximum number of SoAd routing groups. Furthermore it defines the platform type used for RoutingGroupIdType. If SoAdRoutingGroupMax is not greater than 256, an uint8 is used, otherwise an uint16.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	

	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00126 :</b>		
<b>Name</b>	SoAdSoConMax		
<b>Description</b>	Specifies the maximum number of SoAd socket connections. Furthermore it defines the platform type used for SoAd_SoConIdType. If SoAdSoConMax is not greater than 256, an uint8 is used, otherwise uint16.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00004 :</b>		
<b>Name</b>	SoAdVersionInfoApi		
<b>Description</b>	Activates the SoAd_GetVersionInfo() API. TRUE: Enables the SoAd_GetVersionInfo() API. FALSE: SoAd_GetVersionInfo() API is not included.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>No Included Containers</b>
-------------------------------



## 10.2.4 SoAdConfig

<b>SWS Item</b>	<b>ECUC_SoAd_00103 :</b>
<b>Container Name</b>	SoAdConfig
<b>Description</b>	This container contains the configuration parameters and sub containers of the AUTOSAR SoAd module.
<b>Configuration Parameters</b>	

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdPduRoute	0..*	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission. This PDU can consume meta data items of type SOCKET_CONNECTION_ID_16.
SoAdRoutingGroup	0..*	Each container describes a specific routing group which can be enabled or disabled. A routing group consists of PDUs. Routing of PDUs can either be forwarding of PDUs from the upper layer to a TCP or UDP socket of the TCP/IP stack specified by a SoAdPduRoute or the other way around specified by a SoAdSocketRoute.
SoAdSocketConnectionGroup	1..*	Specifies the configuration of a socket connection group, i.e. specifies the socket connections belonging to the group and the parameters which are common for all socket connections of the group. A socket connection specifies how data can be received and transmitted via a TCP or UDP socket.
SoAdSocketRoute	0..*	Describes the path of a PDU from a socket in the TCP/IP stack to an upper layer of the SoAd after reception in the TCP/IP Stack.

## 10.2.5 SoAdSocketConnectionGroup

<b>SWS Item</b>	<b>ECUC_SoAd_00130 :</b>
<b>Container Name</b>	SoAdSocketConnectionGroup
<b>Description</b>	Specifies the configuration of a socket connection group, i.e. specifies the socket connections belonging to the group and the parameters which are common for all socket connections of the group. A socket connection specifies how data can be received and transmitted via a TCP or UDP socket.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00131 :</b>		
<b>Name</b>	SoAdPduHeaderEnable		
<b>Description</b>	Enables the transmission of the PDU header (ID, length) on this socket connection. TRUE: add SoAd PDU header before PDU data FALSE: No SoAd PDU header is used		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>SWS Item</b>	<b>ECUC_SoAd_00110 :</b>		
<b>Name</b>	SoAdSocketAutomaticSoConSetup		
<b>Description</b>	Specifies if the setup of the socket connection shall be done automatically (TRUE) or manually (FALSE) via SoAd_OpenSoCon() and SoAd_CloseSoCon().		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00158 :</b>		
<b>Name</b>	SoAdSocketDifferentiatedServicesField		
<b>Description</b>	The 6-bit Differentiated Service Field in the IP headers may be used for classifying network traffic. If not set a value of zero is used to indicate packets that have not been classified.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 63		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00157 :</b>		
<b>Name</b>	SoAdSocketFlowLabel		
<b>Description</b>	The 20-bit Flow Label field in the IPv6 header may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service. If not set a Flow Label of zero is used to indicate packets that have not been labeled.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 1048575		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

Scope / Dependency		scope: local	
SWS Item		ECUC_SoAd_00138 :	
Name		SoAdSocketFramePriority	
Description		Specifies the priority of the Ethernet frame. If IEEE 802.1Q VLAN Tags are used, the specified priority will be used in the VLAN Tag PCP filed. If this optional parameter is not available the default priority specified in the Tcplp module is used.	
Multiplicity		0..1	
Type		EcucIntegerParamDef	
Range		0 .. 7	
Default value		--	
Post-Build Variant Multiplicity		true	
Post-Build Variant Value		true	
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency		scope: local	

<b>SWS Item</b>	<b>ECUC_SoAd_00112 :</b>		
<b>Name</b>	SoAdSocketIpAddrAssignmentChgNotification		
<b>Description</b>	Specifies if the local IP address assignment change notification callback function of the upper layer shall be called if the assignment of the local IP address used by this socket connection changes.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>		scope: local	

<b>SWS Item</b>	<b>ECUC_SoAd_00018 :</b>		
<b>Name</b>	SoAdSocketLocalPort		
<b>Description</b>	Local UDP or TCP port used for this connection. If this parameter set to 0 SoAd requests Tcplp to select an ephemeral port.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	0		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local		
<b>SWS Item</b>	<b>ECUC_SoAd_00137 :</b>		
<b>Name</b>	SoAdSocketMsgAcceptanceFilterEnabled		
<b>Description</b>	<p>Specifies if the message acceptance filter is enabled (TRUE) or not (FALSE).</p> <p>Note: if a wildcard is used in SoAdSocketRemoteAddress AND SoAdSocketUdpListenOnly is FALSE, this parameter must be TRUE.</p> <p>Note: if multiple SoAdSocketConnections are configured for one SoAdSocketConnectionGroup, this parameter must be TRUE.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdSocketRemoteAddress, SoAdSocketUdpListenOnly, SoAdSocketConnectionGroup		

<b>SWS Item</b>	<b>ECUC_SoAd_00156 :</b>		
<b>Name</b>	SoAdSocketPathMTUEnable		
<b>Description</b>	<p>Specifies if path MTU discovery shall be performed for this connection. If this optional parameter is not available the default behavior configured for the controller in the TcpIp module via the parameter TcpIpV4PathMtuEnabled or TcpIpV6PathMtuEnabled is applied.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity</b>			
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: This parameter must not be set to TRUE if TcpIpV4PathMtuEnabled or TcpIpV6PathMtuEnabled is set to FALSE.		

<b>SWS Item</b>	<b>ECUC_SoAd_00111 :</b>		
<b>Name</b>	SoAdSocketSoConModeChgNotification		
<b>Description</b>	<p>Specifies if the SoCon mode change notification callback function of the upper layer shall be called in case of SoCon mode change.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD



<b>Scope / Dependency</b>		scope: local	
---------------------------	--	--------------	--

<b>SWS Item</b>		<b>ECUC_SoAd_00134 :</b>	
<b>Name</b>		SoAdSocketTpRxBufferMin	
<b>Description</b>		Specifies the amount of data in bytes (PDU data for the upper layer and PDU Header if used) the SoAd shall at least be able to buffer for data reception via each socket connection of the socket connection group and using an upper layer with TP. Note: in case of a TCP socket where PduHeaderMode is used and an upper layer with IF-API, the required buffer size can be determined automatically.	
<b>Multiplicity</b>		0..1	
<b>Type</b>		EcucIntegerParamDef	
<b>Range</b>		0 .. 65535	
<b>Default value</b>		--	
<b>Post-Build Variant Multiplicity</b>		true	
<b>Post-Build Variant Value</b>		true	
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>		scope: local	

<b>SWS Item</b>		<b>ECUC_SoAd_00017 :</b>	
<b>Name</b>		SoAdSocketLocalAddressRef	
<b>Description</b>		Local IP address and interface used for this connection.	
<b>Multiplicity</b>		1	
<b>Type</b>		Symbolic name reference to [ TcpIpLocalAddr ]	
<b>Post-Build Variant Value</b>		true	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>		scope: local	

<b>SWS Item</b>		<b>ECUC_SoAd_00161 :</b>	
<b>Name</b>		SoAdSocketSoConModeChgNotifUpperLayerRef	
<b>Description</b>		Reference to an additional upper layer that shall receive socket connection state changes (although it is not a direct upper layer of the socket connection).	
<b>Multiplicity</b>		0..1	
<b>Type</b>		Reference to [ SoAdBswModules ]	
<b>Post-Build Variant Multiplicity</b>		false	
<b>Post-Build Variant Value</b>		false	
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdSocketConnection	1..*	Specifies the socket connection of the socket connection group.
SoAdSocketProtocol	1	Choice container: transport protocol for socket connection group is either UDP or TCP

### 10.2.6 SoAdSocketConnection

<b>SWS Item</b>	<b>ECUC_SoAd_00009 :</b>		
<b>Container Name</b>	SoAdSocketConnection		
<b>Description</b>	Specifies the socket connection (Id and remote address information). Note: Parameters which are common to all socket connections of a socket connection group are specified directly at the group.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_SoAd_00016 :</b>		
<b>Name</b>	SoAdSocketId		
<b>Description</b>	Socket connection identifier used as SoConId in the interaction with upper layers.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdSocketRemoteAddress	0..1	Container to specify the remote address (IP address and port) for a socket connection. If SoAdSocketRemoteAddress is not specified the remote address has to be set by the upper layer via SoAd_SetRemoteAddr().

## 10.2.7 SoAdSocketProtocol

<b>SWS Item</b>	<b>ECUC_SoAd_00139 :</b>
<b>Choice container Name</b>	SoAdSocketProtocol
<b>Description</b>	Specifies the transport protocol and transport protocol specific parameters used for the socket connections of the socket connection group.

<b>Container Choices</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdSocketTcp	0..1	TCP is used as transport protocol for the socket connection group. Container contains TCP specific parameters.
SoAdSocketUdp	0..1	UDP is used as transport protocol for the socket connection group. Container contains UDP specific parameters.

## 10.2.8 SoAdSocketUdp

<b>SWS Item</b>	<b>ECUC_SoAd_00140 :</b>
<b>Container Name</b>	SoAdSocketUdp
<b>Description</b>	Specifies that UDP is used as transport protocol for the socket connection group and parameters only related to UDP socket connections.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00135 :</b>		
<b>Name</b>	SoAdSocketnPduUdpTxBufferMin		
<b>Description</b>	Specifies the amount of data in bytes (PDU data provided by the upper layer and PDU Header if used) the SoAd shall be able to buffer for data transmission via this socket connection in case the UDP message shall be buffered for transmission of multiple PDUs per UDP. Note: in case of a UDP socket and an upper layer with TP API is configured, the required buffer size can be determined automatically. This optional parameter is only relevant if a nPduUdpTxBuffer is used.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	true	
<b>Post-Build Variant Value</b>		true	
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local dependency: SoAdTxUdpTriggerMode
---------------------------	--

<b>SWS Item</b>	<b>ECUC SoAd 00149 :</b>		
<b>Name</b>	SoAdSocketUdpAliveSupervisionTimeout		
<b>Description</b>	Specifies the time in [s] a UDP socket connection remains in the mode SOAD_SOCON_ONLINE after the latest reception of a frame from the remote peer specified by the remote address. If this optional parameter is not enabled UDP Alive Supervision is deactivated for the related socket connection group.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC SoAd 00159 :</b>		
<b>Name</b>	SoAdSocketUdpChecksumEnabled		
<b>Description</b>	Specifies if UDP checksum calculation shall be enabled (TRUE) or skipped (FALSE) on the related socket. FALSE implies that the upper layer of the socket connection is either capable to handle malformed messages or applies a checksum mechanism itself.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC SoAd 00024 :</b>		
<b>Name</b>	SoAdSocketUdpListenOnly		
<b>Description</b>	Specifies if the socket connection group is only used for reception (TRUE) or used for both reception and transmission (FALSE).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local dependency: SoAdSocketProtocol
---------------------------	--

<b>SWS Item</b>	<b>ECUC SoAd 00154 :</b>		
<b>Name</b>	SoAdSocketUdpStrictHeaderLenCheckEnabled		
<b>Description</b>	Specifies if UDP messages shall be dropped (TRUE) if the length of all contained PDUs does not match the length of the whole message or not (FALSE). Shall only be set to TRUE if SoAdPduHeaderEnable is also set to TRUE.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdPduHeaderEnable		

<b>SWS Item</b>	<b>ECUC SoAd 00133 :</b>		
<b>Name</b>	SoAdSocketUdpTriggerTimeout		
<b>Description</b>	Specifies the timeout in [s] a nPduUdpTxBuffer is waiting for a PDU with TriggerMode = TRIGGER_ALWAYS, i.e. when the timeout expires the nPduUdpTxBuffer is transmitted. Timer is reset after each UDP transmission. This optional parameter is only relevant if a nPduUdpTxBuffer is used.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdTxUdpTriggerMode		

<b>No Included Containers</b>
-------------------------------

### 10.2.9 SoAdSocketTcp

<b>SWS Item</b>	<b>ECUC_SoAd_00141 :</b>
<b>Container Name</b>	SoAdSocketTcp
<b>Description</b>	Specifies that TCP is used as transport protocol for the socket connection group and parameters only related to TCP socket connections.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00147 :</b>		
<b>Name</b>	SoAdSocketTcpImmediateTpTxConfirmation		
<b>Description</b>	If set to FALSE, SoAd notifies the TP upper layer via transmit confirmation after a Tcp Ack has been received. If set to TRUE, SoAd notifies the TP upper layer via transmit confirmation immediately after transmit has been accepted by TcpIp.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00022 :</b>		
<b>Name</b>	SoAdSocketTcpInitiate		
<b>Description</b>	Specifies the initiator for this TCP connection. It will not be defined for UDP sockets. TRUE: This TCP connection is initiated by this module. FALSE: This TCP connection is to be initiated in the listen mode.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00148 :</b>		
<b>Name</b>	SoAdSocketTcpKeepAlive		
<b>Description</b>	Specifies to use the keep-alive mechanism for this connection. It will not be defined for UDP sockets. TRUE: This TCP connection will use the keep-alive mechanism. FALSE: This TCP connection will not use the keep-alive mechanism. Note: This parameter must not be set to TRUE if TcpIpTcpKeepAliveEnabled is set to FALSE.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local dependency: TcpIpTcpKeepAliveEnabled
---------------------------	--

<b>SWS Item</b>	<b>ECUC SoAd 00152 :</b>		
<b>Name</b>	SoAdSocketTcpKeepAliveInterval		
<b>Description</b>	Specifies the interval in seconds between subsequent keepalive probes.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdSocketTcpKeepAlive		

<b>SWS Item</b>	<b>ECUC SoAd 00151 :</b>		
<b>Name</b>	SoAdSocketTcpKeepAliveProbesMax		
<b>Description</b>	Maximum number of times that TCP retransmits an individual data segment before aborting the connection.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdSocketTcpKeepAlive		

<b>SWS Item</b>	<b>ECUC SoAd 00153 :</b>		
<b>Name</b>	SoAdSocketTcpKeepAliveTime		
<b>Description</b>	Specifies the time in seconds between the last data packet sent and the first keepalive probe.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	7200		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD



<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdSocketTcpKeepAlive		

<b>SWS Item</b>	<b>ECUC_SoAd_00023 :</b>		
<b>Name</b>	SoAdSocketTcpNoDelay		
<b>Description</b>	Specifies not to use the congestion control mechanism for this connection. It will not be defined for UDP sockets. TRUE: This TCP connection will NOT use congestion control. FALSE: This TCP connection will use congestion control. If the optional parameter is not enabled, the default behavior configured for TcpIp via the parameter TcpIpTcpNagleEnabled is applied. Note: This parameter must not be set to FALSE if TcpIpTcpNagleEnabled is set to FALSE.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: TcpIpTcpNagleEnabled		

<b>SWS Item</b>	<b>ECUC_SoAd_00142 :</b>		
<b>Name</b>	SoAdSocketTcpTxQuota		
<b>Description</b>	Specifies the maximum amount of bytes (PDU data provided by the upper layer and PDU Header if used) the SoAd may queue for transmission via TCP at the TcpIp module for each socket connection of this socket connection group. Rationale: prohibits that a socket connection consumes all available transmit buffers at the TcpIp and blocks transmissions via other socket connections. If the optional parameter is not enabled, the amount of data is not limited.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>		scope: local	
<b>SWS Item</b>		<b>ECUC_SoAd_00155 :</b>	
<b>Name</b>		SoAdSocketTCPOptionFilterRef	
<b>Description</b>		Specifies which TCP option filter shall be applied on the related socket.	
<b>Multiplicity</b>		0..1	
<b>Type</b>		Symbolic name reference to [ TcpIpTcpConfigOptionFilter ]	
<b>Post-Build Multiplicity</b>		<b>Variant</b> true	
<b>Post-Build Variant Value</b>		true	
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			
<b>No Included Containers</b>			

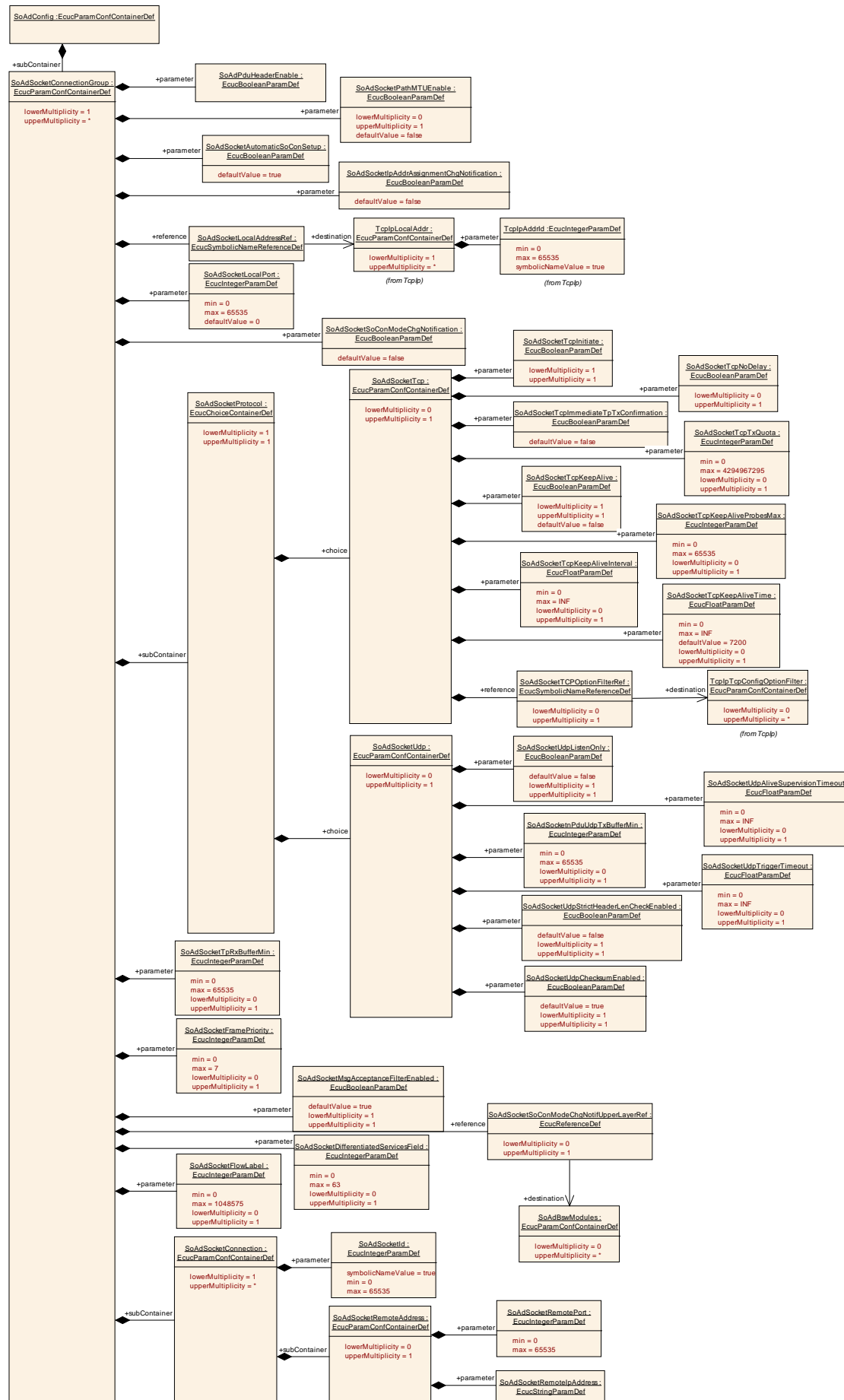


Figure 4: Socket Connection Group configuration

### 10.2.10 SoAdSocketRemoteAddress

<b>SWS Item</b>	<b>ECUC_SoAd_00113 :</b>
<b>Container Name</b>	SoAdSocketRemoteAddress
<b>Description</b>	Subcontainer of SoAdSocketConnection to specify the remote address (IP address and port) for a socket connection. If SoAdSocketRemoteAddress is not specified the remote address has to be set by the upper layer via SoAd_SetRemoteAddr().
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00019 :</b>		
<b>Name</b>	SoAdSocketRemotelpAddress		
<b>Description</b>	IP address of remote node. The configured address must be of the same TcplpDomainType (i.e. IPv4 or IPv6) as the TcplpLocalAddr referred by SoAdSocketLocalAddressRef . To accept any remote IP address, set SoAdSocketRemotelpAddress to "ANY". See message acceptance policy for more details.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucStringParamDef		
<b>Default value</b>	--		
<b>maxLength</b>	--		
<b>minLength</b>	--		
<b>regularExpression</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: TcplpDomainType		

<b>SWS Item</b>	<b>ECUC_SoAd_00020 :</b>		
<b>Name</b>	SoAdSocketRemotePort		
<b>Description</b>	Remote UDP or TCP port used for this connection. To accept any remote port, set SoAdSocketRemotePort to 0. See message acceptance policy for more details.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

## 10.2.11 SoAdSocketRoute

<b>SWS Item</b>	<b>ECUC_SoAd_00008 :</b>
<b>Container Name</b>	SoAdSocketRoute
<b>Description</b>	Describes the path of a PDU from a socket in the TCP/IP stack to an upper layer of the SoAd after reception in the TCP/IP Stack.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00036 :</b>		
<b>Name</b>	SoAdRxPduHeaderId		
<b>Description</b>	ID contained in the packet received on the TCP/IP connection if the PDU header option is enabled.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967296		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: Parameter shall be configured for SoAdSocketRoute that is related to a SoAdSocketConnection belonging to a SoAdSocketConnectionGroup with SoAdPduHeaderEnable set to TRUE.		

<b>SWS Item</b>	<b>ECUC_SoAd_00035 :</b>		
<b>Name</b>	SoAdRxSocketConnOrSocketConnBundleRef		
<b>Description</b>	Choice Reference to a SocketConnection or to a SocketConnectionGroup on which the PDU was received. The reference to a SocketConnectionGroup shall only be used for upper layers with IF API.		
<b>Multiplicity</b>	1		
<b>Type</b>	Choice reference to [ SoAdSocketConnection , SoAdSocketConnectionGroup ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdRxUpperLayerType		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdSocketRouteDest	1	Container which specifies the socket route destination.

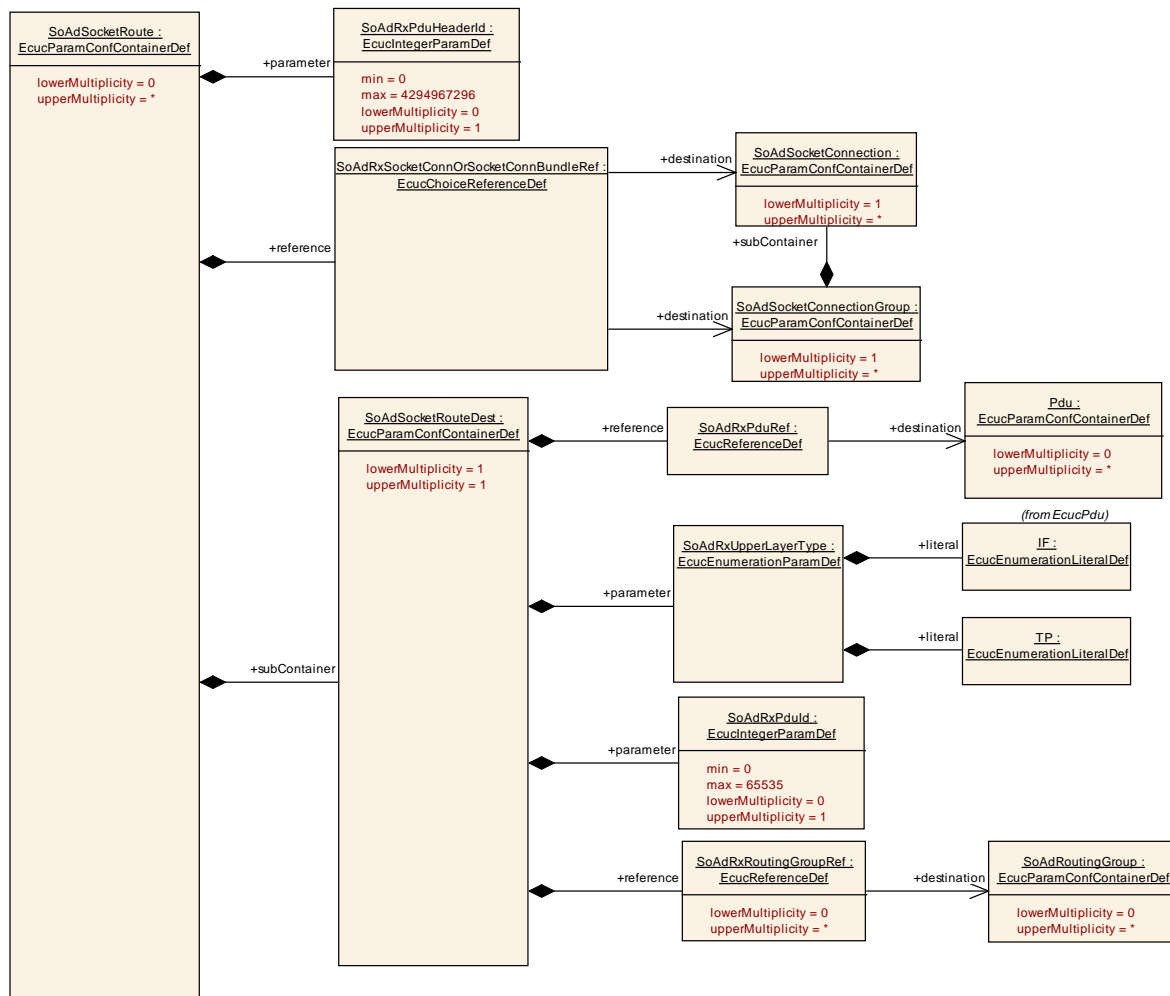


Figure 5: Socket Route configuration

## 10.2.12 SoAdSocketRouteDest

<b>SWS Item</b>	<b>ECUC_SoAd_00114 :</b>
<b>Container Name</b>	SoAdSocketRouteDest
<b>Description</b>	Describes the upper layer destination PDU for a message received on a TcpIp socket. This PDU can produce meta data items of type SOCKET_CONNECTION_ID_16.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00116 :</b>		
<b>Name</b>	SoAdRxPduld		
<b>Description</b>	This unique identifier is used for a receive cancellation request from an upper layer of the SoAd.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00115 :</b>		
<b>Name</b>	SoAdRxUpperLayerType		
<b>Description</b>	Specifies the upper layer interface type (must be "IF" in case of multiple RxPdus).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	IF		If interface
	TP		TP interface
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00038 :</b>		
<b>Name</b>	SoAdRxPduRef		
<b>Description</b>	Reference to the global PDU structure		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

Scope / Dependency		scope: ECU	
SWS Item		ECUC_SoAd_00117 :	
Name		SoAdRxRoutingGroupRef	
Description		Reference to the routing group.	
Multiplicity		0..*	
Type		Reference to [ SoAdRoutingGroup ]	
Post-Build Variant Multiplicity		true	
Post-Build Variant Value		true	
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency		scope: local	

**No Included Containers**

### 10.2.13 SoAdPduRoute

<b>SWS Item</b>	<b>ECUC_SoAd_00007 :</b>
<b>Container Name</b>	SoAdPduRoute
<b>Description</b>	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission. This PDU can consume meta data items of type SOCKET_CONNECTION_ID_16.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00160 :</b>		
<b>Name</b>	SoAdTxPduCollectionSemantics		
<b>Description</b>	Specifies if this PDU shall be collected using a queued or last-is-best semantics. This parameter is only relevant if the PDU collection feature is enabled. Shall only be set to SOAD_COLLECT_LAST_IS_BEST if the related upper layer is configured with SoAdIfTriggerTransmit set to TRUE.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	SOAD_COLLECT_LAST_IS_BEST	The PDU data will be fetched via <Up>_[SoAd][If]TriggerTransmit just before the transmission executes.	
	SOAD_COLLECT_QUEUED	The PDU data will instantly be stored in the context of the SoAd_IfTransmit API.	
<b>Default value</b>	SOAD_COLLECT_QUEUED		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD



<b>Scope</b>	/scope: local
<b>Dependency</b>	dependency: SoAdSocketnPduUdpTxBufferMin, SoAdIfTriggerTransmit

<b>SWS Item</b>	<b>ECUC_SoAd_00031 :</b>		
<b>Name</b>	SoAdTxPduId		
<b>Description</b>	Tx PDU ID of the PDU coming from the PDU Router.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

SWS Item	ECUC_SoAd_00118 :		
Name	SoAdTxUpperLayerType		
Description	Specifies the upper layer interface type (must be "IF" in case of multiple PduRoutes).		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	IF	If Interface	
	TP	TP Interface	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope Dependency	/scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00030 :</b>		
<b>Name</b>	SoAdTxPduRef		
<b>Description</b>	Reference to the global PDU structure		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
SoAdPduRouteDest	1..*	Container which specifies the PDU route destination.

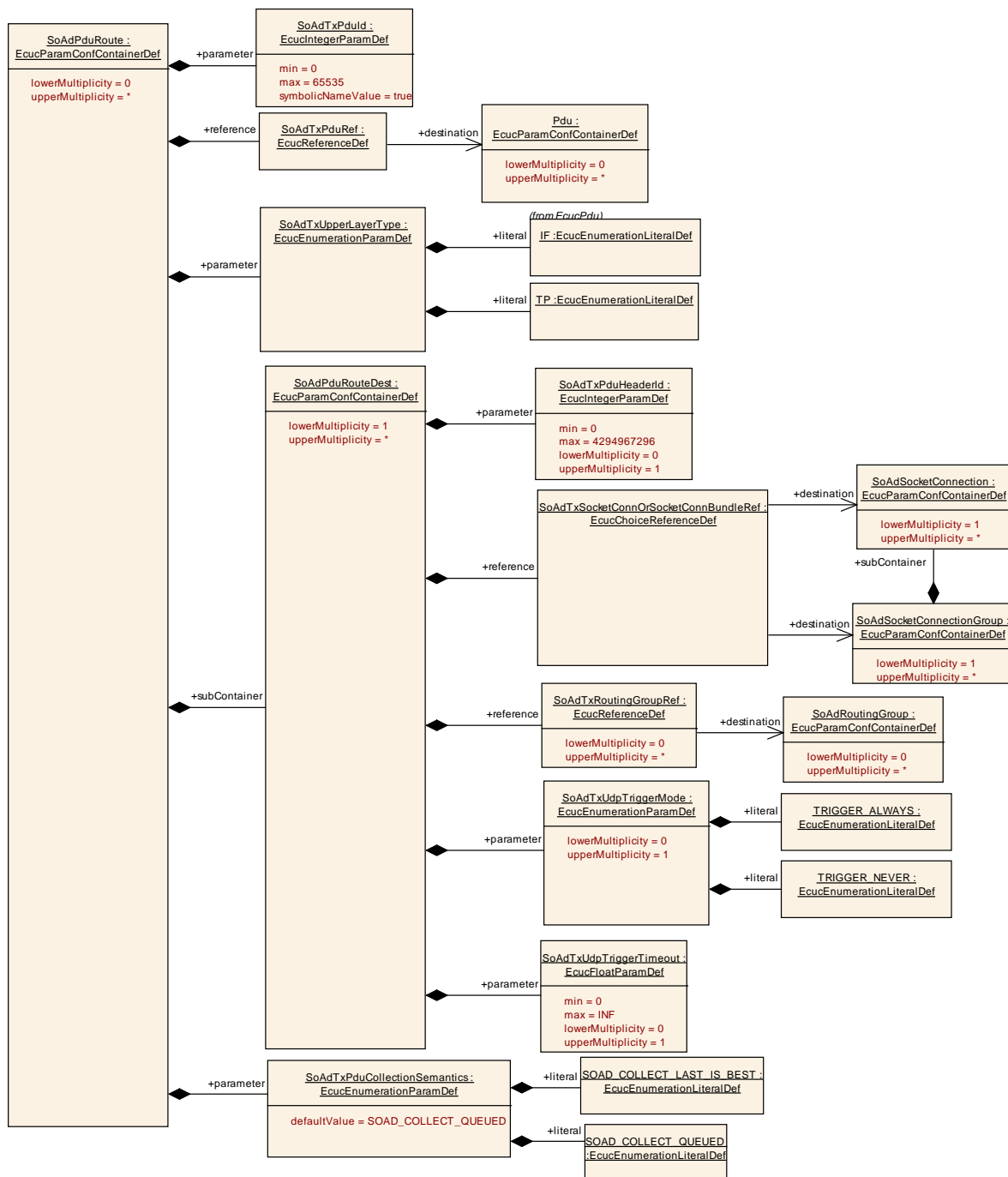


Figure 6: Pdu Route configuration

## 10.2.14 SoAdPduRouteDest

<b>SWS Item</b>	<b>ECUC_SoAd_00119 :</b>
<b>Container Name</b>	SoAdPduRouteDest
<b>Description</b>	Specifies the PDU route destination.
<b>Configuration Parameters</b>	

SWS Item	ECUC_SoAd_00120 :		
Name	SoAdTxPduHeaderId		
Description	ID to be sent on the TCP/IP connection if the PDU header option is enabled.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967296		
Default value	--		
Post-Build Multiplicity	Variant	true	
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: Parameter shall be configured for SoAdPduRouteDest that is related to a SoAdSocketConnection belonging to a SoAdSocketConnectionGroup with SoAdPduHeaderEnable set to TRUE.		

<b>SWS Item</b>	<b>ECUC_SoAd_00136 :</b>		
<b>Name</b>	SoAdTxUdpTriggerMode		
<b>Description</b>	Specifies whether a PDU triggers the transmission of the nPduUdpTxBuffer. If this parameter is set to TRIGGER_NEVER, SoAd shall use an nPduUdpTxBuffer for the related socket connection. nPduUdpTxBuffer can only be used for upper layers with IF API, i.e. this parameter shall only be set to TRIGGER_NEVER if all upper layers belonging to the related socket connection have SoAdTxUpperLayerType set to "IF". This parameter is only relevant for UDP connections.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	TRIGGER_ALWAYS	PDU triggers the transmission	
	TRIGGER_NEVER	PDU does not trigger the transmission	
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope</b>	scope: local
<b>Dependency</b>	dependency: SoAdSocketProtocol, SoAdTxUpperLayerType

<b>SWS Item</b>	<b>ECUC_SoAd_00150 :</b>		
<b>Name</b>	SoAdTxUdpTriggerTimeout		
<b>Description</b>	Specifies the timeout in [s] the nPduUdpTxBuffer shall be transmitted at the latest after this PDU is put into the buffer. This optional parameter is only relevant if SoAdTxUdpTriggerMode is TRIGGER_NEVER.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	true	
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: SoAdTxUdpTriggerMode		

<b>SWS Item</b>	<b>ECUC_SoAd_00123 :</b>		
<b>Name</b>	SoAdTxRoutingGroupRef		
<b>Description</b>	Reference to the routing group.		
<b>Multiplicity</b>	0..*		
<b>Type</b>	Reference to [ SoAdRoutingGroup ]		
<b>Post-Build Multiplicity</b>	<b>Variant</b>	true	
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC SoAd_00034 :</b>		
<b>Name</b>	SoAdTxSocketConnOrSocketConnBundleRef		
<b>Description</b>	Choice Reference to a SocketConnection or to a SocketConnectionGroup on which the PDU is to be sent on. The reference to a SocketConnectionGroup shall only be used for upper layers with IF API.		
<b>Multiplicity</b>	1		
<b>Type</b>	Choice reference to [ SoAdSocketConnection , SoAdSocketConnectionGroup ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local dependency: SoAdTxUpperLayerType
---------------------------	--

#### No Included Containers

### 10.2.15 SoAdRoutingGroup

<b>SWS Item</b>	<b>ECUC_SoAd_00109 :</b>
<b>Container Name</b>	SoAdRoutingGroup
<b>Description</b>	Each container describes a specific routing group which can be enabled or disabled. A routing group consists of PDUs. Routing of PDUs can either be forwarding of PDUs from the upper layer to a TCP or UDP socket of the TCP/IP stack specified by a SoAdPduRoute or the other way around specified by a SoAdSocketRoute.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_SoAd_00121 :</b>		
<b>Name</b>	SoAdRoutingGroupId		
<b>Description</b>	Unique ID of Routing Group		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>ECUC_SoAd_00122 :</b>		
<b>Name</b>	SoAdRoutingGroupsEnabledAtInit		
<b>Description</b>	If set to true this routing group will be enabled after initializing the SoAd module (i.e. enabled in the SoAd_Init function).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_SoAd_00146 :</b>		
<b>Name</b>	SoAdRoutingGroupTxTriggerable		
<b>Description</b>	Specifies if the If-TxPDUs related to the PduRouteDest containers referenced by this routing group can be triggered via SoAd_IfRoutingGroupTransmit (TRUE) or not (FALSE).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	

	Post-build time	--	
Scope / Dependency	scope: local		
No Included Containers			

### **10.3 Published Information**

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.

## 11 Not applicable requirements

**[SWS\_SoAd\_00296]** [These requirements are not applicable to this specification.]  
(SRS\_BSW\_00170, SRS\_BSW\_00375, SRS\_BSW\_00416, SRS\_BSW\_00168,  
SRS\_BSW\_00423, SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_BSW\_00426,  
SRS\_BSW\_00427, SRS\_BSW\_00429, SRS\_BSW\_00432, SRS\_BSW\_00336,  
SRS\_BSW\_00417, SRS\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00005,  
SRS\_BSW\_00415, SRS\_BSW\_00164, SRS\_BSW\_00325, SRS\_BSW\_00160,  
SRS\_BSW\_00413, SRS\_BSW\_00347, SRS\_BSW\_00307, SRS\_BSW\_00335,  
SRS\_BSW\_00410, SRS\_BSW\_00314, SRS\_BSW\_00328, SRS\_BSW\_00312,  
SRS\_BSW\_00006, SRS\_BSW\_00306, SRS\_BSW\_00309, SRS\_BSW\_00330,  
SRS\_BSW\_00331, SRS\_BSW\_00172, SRS\_BSW\_00010, SRS\_BSW\_00333,  
SRS\_BSW\_00321, SRS\_BSW\_00341, SRS\_BSW\_00334)