

Document Title	Specification of Socket Adaptor
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	416
Document Classification	Standard
Document Version	2.2.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History			
Date	Version	Changed by	Change Description
31.03.2014	2.2.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • TP API: Harmonization of ChangeParameter function • Clarifications and corrections of requirements and sequence charts • Editorial changes
31.10.2013	2.1.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • TP API: NotifResultType replaced by Std_ReturnType • Clarifications and corrections of requirements and sequence charts • Editorial changes • Removed chapter(s) on change documentation
26.02.2013	2.0.0	AUTOSAR Administration	<ul style="list-style-type: none"> • APIs for SWS TCP/IP module interaction added/updated • Functionality to trigger IPdus for sending has been added • DoIP functionality removed
03.11.2011	1.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Rectify inconsistencies in synchronicity and reentrancy • Adjust parameter multiplicity • New traceability mechanism
14.10.2010	1.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> • ComStack Harmonization. • Allow for Post-Build Configuration • API for IP address change notification • Allow full handling of TCP connections
16.12.2009	1.0.0	AUTOSAR Administration	Initial Release Revision

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	6
2	Acronyms and abbreviations	8
3	Related documentation.....	9
3.1	Input documents.....	9
3.2	Related standards and norms	10
3.3	Related specification	10
4	Constraints and assumptions	11
4.1	Limitations.....	11
4.2	Applicability to car domains.....	11
5	Dependencies on other modules.....	12
5.1	AUTOSAR TCP/IP Stack	12
5.2	Generic Upper Layer	12
5.3	File structure	12
5.3.1	Code file structure.....	12
5.3.2	Header file structure.....	12
5.4	Version check.....	13
6	Requirements traceability	14
7	Functional specification	22
7.1	Socket Connections	22
7.1.1	Socket Connection Open	23
7.1.2	Socket Connection Close.....	26
7.1.3	Socket Connection Open/Close Sequence Remarks.....	28
7.1.4	Notifications.....	29
7.2	PDU Transmission	30
7.2.1	PDU Transmission via IF-API.....	30
7.2.2	PDU Transmission via IF-API and nPduUdpTxBuffer	31
7.2.3	PDU Transmission via IfRoutingGroupTransmit API.....	33
7.2.4	PDU Transmission via TP-API	33
7.3	PDU Header option	36
7.4	PDU Reception.....	36
7.4.1	PDU Reception via IF-API.....	37
7.4.2	PDU Reception via TP-API (PDU Header disabled)	38
7.4.3	PDU Reception via TP-API (PDU Header enabled)	39
7.5	Best Match Algorithm	40
7.6	Message Acceptance Policy	40
7.7	TP PDU Cancelation	41
7.8	Disconnection and recovery	42
7.9	Routing Groups	42
7.10	PDU fan-out	43
7.11	Resource Management Option	44
7.12	Buffer handling	46
7.13	Error Handling	46
7.13.1	Error classification.....	46
7.13.2	Error detection.....	47

7.13.3	Error notification	47
7.14	Application notes	47
7.15	Debugging Concept.....	47
7.16	Version checking	47
8	API specification	48
8.1	Imported types.....	48
8.2	Type definitions	48
8.2.1	SoAd_SoConIdType	48
8.2.2	SoAd_SoConModeType	49
8.2.3	SoAd_RoutingGroupIdType	49
8.2.4	SoAd_ConfigType	49
8.3	Function definitions	50
8.3.1	General	50
8.3.2	Normal Operation.....	52
8.3.3	Transmit/Receive Cancellation API.....	55
8.3.4	Information and Control API.....	56
8.4	Call-back notifications	71
8.4.1	SoAd_RxIndication.....	71
8.4.2	SoAd_CopyTxData	72
8.4.3	SoAd_TxConfirmation	73
8.4.4	SoAd_TcpAccepted	74
8.4.5	SoAd_TcpConnected.....	75
8.4.6	SoAd_TcpIpEvent	75
8.4.7	SoAd_LocalIpAddrAssignmentChg.....	76
8.5	Scheduled functions	77
8.5.1	Terms and definitions.....	77
8.5.2	SoAd_MainFunction.....	77
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	86
9.1	Address – Assignment	86
9.2	Socket Connection Setup – UDP	87
9.3	Socket Connection Setup – TCP.....	89
9.4	Reception – Upper Layer If API.....	91
9.5	Reception – Upper Layer TP API	92
9.6	Transmission – Upper Layer If API – TCP	94
9.7	Transmission – Upper Layer If API – UDP	96
9.8	Transmission – Upper Layer TP API	97
10	Configuration specification	99
10.1	How to read this chapter	99
10.2	Containers and configuration parameters	100
10.2.1	Variants.....	100
10.2.2	SoAd	101
10.2.3	SoAdBswModules	101
10.2.4	SoAdGeneral.....	104
10.2.5	SoAdConfig.....	106

10.2.6	SoAdSocketConnectionGroup	107
10.2.7	SoAdSocketConnection	111
10.2.8	SoAdSocketProtocol	111
10.2.9	SoAdSocketUdp.....	112
10.2.10	SoAdSocketTcp	113
10.2.11	SoAdSocketRemoteAddress.....	117
10.2.12	SoAdSocketRoute.....	118
10.2.13	SoAdSocketRouteDest	119
10.2.14	SoAdPduRoute	121
10.2.15	SoAdPduRouteDest.....	122
10.2.16	SoAdRoutingGroup	124
10.3	Published Information.....	125
11	Not applicable requirements	126

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 haven been applied between AUTOSAR R4.0.3 and AUTOSAR R4.1.1.

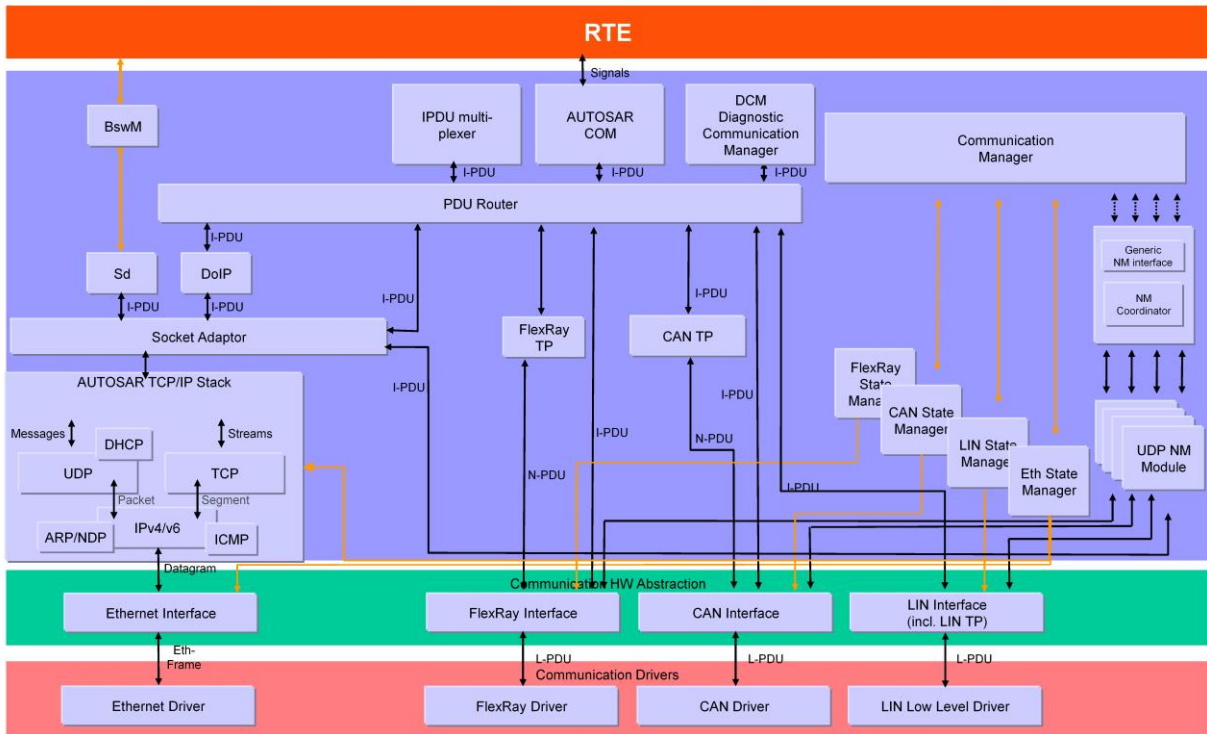


Figure 1: Extended AUTOSAR Communication Stack.

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
ARP	Address Resolution Protocol
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DHCP	Dynamic Host Configuration Protocol
Sd	Service Discovery
DoIP	Diagnostics over IP
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
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

Term:	Description:
AUTOSAR Connector	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.

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 Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.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

none

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, IP, ARP, ICMP, DHCP) and provides dynamic, socket based communication via Ethernet. The SoAd module is the sole upper layer module 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]
- _Cbk.h (for callback functions of the SoAd upper layer modules)] ()

[SWS_SoAd_00070] [The SoAd module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.] ()

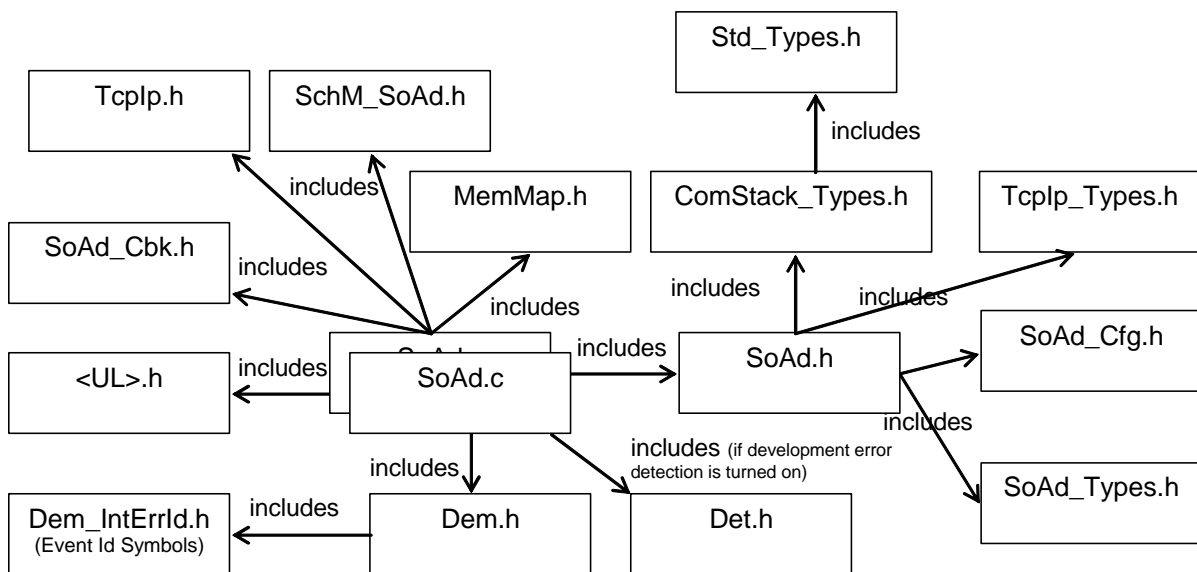


Figure 2: AUTOSAR SoAd header file structure.

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
-	-	SWS_SoAd_00001
-	-	SWS_SoAd_00070
-	-	SWS_SoAd_00072
-	-	SWS_SoAd_00073
-	-	SWS_SoAd_00091
-	-	SWS_SoAd_00093
-	-	SWS_SoAd_00096
-	-	SWS_SoAd_00097
-	-	SWS_SoAd_00098
-	-	SWS_SoAd_00099
-	-	SWS_SoAd_00100
-	-	SWS_SoAd_00101
-	-	SWS_SoAd_00105
-	-	SWS_SoAd_00106
-	-	SWS_SoAd_00107
-	-	SWS_SoAd_00121
-	-	SWS_SoAd_00125
-	-	SWS_SoAd_00126
-	-	SWS_SoAd_00131
-	-	SWS_SoAd_00137
-	-	SWS_SoAd_00138
-	-	SWS_SoAd_00139
-	-	SWS_SoAd_00146
-	-	SWS_SoAd_00176
-	-	SWS_SoAd_00180
-	-	SWS_SoAd_00181
-	-	SWS_SoAd_00197
-	-	SWS_SoAd_00198
-	-	SWS_SoAd_00208
-	-	SWS_SoAd_00209
-	-	SWS_SoAd_00210
-	-	SWS_SoAd_00211
-	-	SWS_SoAd_00212
-	-	SWS_SoAd_00213
-	-	SWS_SoAd_00214
-	-	SWS_SoAd_00216

-	-	SWS_SoAd_00224
-	-	SWS_SoAd_00237
-	-	SWS_SoAd_00264
-	-	SWS_SoAd_00267
-	-	SWS_SoAd_00268
-	-	SWS_SoAd_00269
-	-	SWS_SoAd_00270
-	-	SWS_SoAd_00271
-	-	SWS_SoAd_00272
-	-	SWS_SoAd_00273
-	-	SWS_SoAd_00274
-	-	SWS_SoAd_00275
-	-	SWS_SoAd_00276
-	-	SWS_SoAd_00277
-	-	SWS_SoAd_00278
-	-	SWS_SoAd_00279
-	-	SWS_SoAd_00280
-	-	SWS_SoAd_00283
-	-	SWS_SoAd_00503
-	-	SWS_SoAd_00504
-	-	SWS_SoAd_00505
-	-	SWS_SoAd_00506
-	-	SWS_SoAd_00507
-	-	SWS_SoAd_00508
-	-	SWS_SoAd_00509
-	-	SWS_SoAd_00510
-	-	SWS_SoAd_00511
-	-	SWS_SoAd_00512
-	-	SWS_SoAd_00513
-	-	SWS_SoAd_00514
-	-	SWS_SoAd_00515
-	-	SWS_SoAd_00516
-	-	SWS_SoAd_00517
-	-	SWS_SoAd_00518
-	-	SWS_SoAd_00519
-	-	SWS_SoAd_00520
-	-	SWS_SoAd_00521
-	-	SWS_SoAd_00522
-	-	SWS_SoAd_00523

-	-	SWS_SoAd_00524
-	-	SWS_SoAd_00525
-	-	SWS_SoAd_00527
-	-	SWS_SoAd_00528
-	-	SWS_SoAd_00529
-	-	SWS_SoAd_00531
-	-	SWS_SoAd_00532
-	-	SWS_SoAd_00533
-	-	SWS_SoAd_00536
-	-	SWS_SoAd_00538
-	-	SWS_SoAd_00539
-	-	SWS_SoAd_00540
-	-	SWS_SoAd_00541
-	-	SWS_SoAd_00542
-	-	SWS_SoAd_00543
-	-	SWS_SoAd_00544
-	-	SWS_SoAd_00545
-	-	SWS_SoAd_00546
-	-	SWS_SoAd_00547
-	-	SWS_SoAd_00548
-	-	SWS_SoAd_00549
-	-	SWS_SoAd_00550
-	-	SWS_SoAd_00551
-	-	SWS_SoAd_00552
-	-	SWS_SoAd_00553
-	-	SWS_SoAd_00554
-	-	SWS_SoAd_00555
-	-	SWS_SoAd_00556
-	-	SWS_SoAd_00557
-	-	SWS_SoAd_00558
-	-	SWS_SoAd_00559
-	-	SWS_SoAd_00560
-	-	SWS_SoAd_00561
-	-	SWS_SoAd_00562
-	-	SWS_SoAd_00563
-	-	SWS_SoAd_00564
-	-	SWS_SoAd_00565
-	-	SWS_SoAd_00566
-	-	SWS_SoAd_00567

-	-	SWS_SoAd_00568
-	-	SWS_SoAd_00569
-	-	SWS_SoAd_00570
-	-	SWS_SoAd_00571
-	-	SWS_SoAd_00572
-	-	SWS_SoAd_00573
-	-	SWS_SoAd_00574
-	-	SWS_SoAd_00575
-	-	SWS_SoAd_00576
-	-	SWS_SoAd_00577
-	-	SWS_SoAd_00581
-	-	SWS_SoAd_00582
-	-	SWS_SoAd_00586
-	-	SWS_SoAd_00587
-	-	SWS_SoAd_00588
-	-	SWS_SoAd_00589
-	-	SWS_SoAd_00590
-	-	SWS_SoAd_00591
-	-	SWS_SoAd_00592
-	-	SWS_SoAd_00593
-	-	SWS_SoAd_00594
-	-	SWS_SoAd_00595
-	-	SWS_SoAd_00597
-	-	SWS_SoAd_00598
-	-	SWS_SoAd_00599
-	-	SWS_SoAd_00600
-	-	SWS_SoAd_00601
-	-	SWS_SoAd_00602
-	-	SWS_SoAd_00604
-	-	SWS_SoAd_00605
-	-	SWS_SoAd_00606
-	-	SWS_SoAd_00607
-	-	SWS_SoAd_00608
-	-	SWS_SoAd_00609
-	-	SWS_SoAd_00610
-	-	SWS_SoAd_00611
-	-	SWS_SoAd_00612
-	-	SWS_SoAd_00613
-	-	SWS_SoAd_00615

-	-	SWS_SoAd_00616
-	-	SWS_SoAd_00617
-	-	SWS_SoAd_00618
-	-	SWS_SoAd_00619
-	-	SWS_SoAd_00620
-	-	SWS_SoAd_00621
-	-	SWS_SoAd_00622
-	-	SWS_SoAd_00623
-	-	SWS_SoAd_00624
-	-	SWS_SoAd_00625
-	-	SWS_SoAd_00626
-	-	SWS_SoAd_00627
-	-	SWS_SoAd_00628
-	-	SWS_SoAd_00629
-	-	SWS_SoAd_00630
-	-	SWS_SoAd_00631
-	-	SWS_SoAd_00632
-	-	SWS_SoAd_00633
-	-	SWS_SoAd_00635
-	-	SWS_SoAd_00636
-	-	SWS_SoAd_00637
-	-	SWS_SoAd_00638
-	-	SWS_SoAd_00639
-	-	SWS_SoAd_00640
-	-	SWS_SoAd_00641
-	-	SWS_SoAd_00642
-	-	SWS_SoAd_00643
-	-	SWS_SoAd_00644
-	-	SWS_SoAd_00645
-	-	SWS_SoAd_00646
-	-	SWS_SoAd_00647
-	-	SWS_SoAd_00648
-	-	SWS_SoAd_00649
-	-	SWS_SoAd_00650
-	-	SWS_SoAd_00651
-	-	SWS_SoAd_00652
-	-	SWS_SoAd_00653
-	-	SWS_SoAd_00655
-	-	SWS_SoAd_00656

-	-	SWS_SoAd_00657
-	-	SWS_SoAd_00658
-	-	SWS_SoAd_00659
-	-	SWS_SoAd_00660
-	-	SWS_SoAd_00661
-	-	SWS_SoAd_00662
-	-	SWS_SoAd_00663
-	-	SWS_SoAd_00664
-	-	SWS_SoAd_00665
-	-	SWS_SoAd_00666
-	-	SWS_SoAd_00667
-	-	SWS_SoAd_00668
-	-	SWS_SoAd_00669
-	-	SWS_SoAd_00670
-	-	SWS_SoAd_00671
-	-	SWS_SoAd_00672
-	-	SWS_SoAd_00673
-	-	SWS_SoAd_00675
-	-	SWS_SoAd_00676
-	-	SWS_SoAd_00678
-	-	SWS_SoAd_00679
-	-	SWS_SoAd_00680
-	-	SWS_SoAd_00681
-	-	SWS_SoAd_00683
-	-	SWS_SoAd_00684
-	-	SWS_SoAd_00685
-	-	SWS_SoAd_00686
-	-	SWS_SoAd_00687
-	-	SWS_SoAd_00688
-	-	SWS_SoAd_00689
-	-	SWS_SoAd_00690
-	-	SWS_SoAd_00691
-	-	SWS_SoAd_00692
BSW00434	-	SWS_SoAd_00296
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_SoAd_00296
SRS_BSW_00006	The source code of software modules above the æC 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	SWS_SoAd_00296

	platforms.	
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 and platform independent	SWS_SoAd_00296
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_00326	-	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_00341	Module documentation shall contains 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_00355	-	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

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.

Conventions:

- A socket connection which uses TCP as transport protocol is called a TCP socket connection (choice container SoAdSocketProtocol contains a SoAdSocketTcp subcontainer).
- A socket connection which uses UDP as transport protocol is called a UDP socket connection (choice container SoAdSocketProtocol contains a SoAdSocketUdp subcontainer).
- An IF-PDU is a PDU which is send/received via the IF-API of the SoAd
- An TP-PDU is a PDU which is send/received via the TP-API of the SoAd

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.⌋()

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 TcpIp 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 `SoAdSocketTcpInitiate` set to TRUE:

- (1) Get an appropriate socket from TcpIp by calling `TcpIp_GetSocket()` with the domain type implicitly specified by `SoAdLocalAddressRef`, 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 `SoAdLocalAddressRef` and local port specified by `SoAdSocketLocalPort`.
- (4) In case of a TCP socket initiate the TCP connection by calling `TcpIp_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_GetSocket()` with the domain type implicitly specified by `SoAdLocalAddressRef`, 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 `SoAdLocalAddressRef` 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_GetSocket()` with the domain type implicitly specified by `SoAdLocalAddressRef`, 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 `SoAdLocalAddressRef` 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`()

[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) SoAdSocketTcplInitiate is set to TRUE

┆()

[SWS_SoAd_00594]┆ At SoAd_TcpAccpeted(), SoAd shall perform the following actions if the TCP SoAdSocketConnectionGroup related to SocketId has SoAdSocketTcplInitiate 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 Tcplp 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 SoAdSocketTcplInitiate and SoAdSocketMsgAcceptanceFilterEnabled set to FALSE and is not online (i.e. current connection state not SOAD_SOCON_ONLINE):

- (1) assign the Tcplp 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 SOAD_SOCON_ONLINE
- (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()`)

J()

[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.

J()

[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 could be transmitted, and
- (2) `E_NOT_OK` for any other cause.

J()

[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 could be delivered to the upper layer, and
- (2) `E_NOT_OK` for any other cause.

J()

[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_TcplpEvent()` with `Event` set to `TCPIP_UDP_CLOSED`, `SoAd` shall

- (1) remove the assignment of the `Tcplp` 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 `Tcplp` sockets)]()

[SWS_SoAd_00645] Within `SoAd_TcplpEvent()` with `Event` set to `TCPIP_TCP_CLOSED` for a Listen-Socket, `SoAd` shall remove the assignment of the `Tcplp` socket identified by `SocketId` from the related TCP socket connection group.]()

[SWS_SoAd_00646] Within `SoAd_TcplpEvent()` with `Event` set to `TCPIP_TCP_CLOSED` or `TCPIP_TCP_RESET`, `SoAd` shall

- (1) remove the assignment of the `Tcplp` 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 `Tcplp` 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_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>_LocalIpAddressAssignmentChg() and the new address state if SoAdSocketIpAddressAssignmentChgNotification 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 SoAdSrcPduId 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_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_00541] If TcpIp_UdpTransmit() returns with TCPIP_E_ARP_CACHE_MISS and SoAdSocketUdpRetryEnabled is set to TRUE, SoAd shall copy the data provided by SoAd_IfTransmit() to a socket connection specific UdpTxBuffer and retry the transmission once per SoAd_MainFunction() call until TcpIp_UdpTransmit() for this socket connection returns with a value other than TCPIP_E_ARP_CACHE_MISS.]()

Note: if a new PDU is transmitted, TcpIp_UdpTransmit() is called with the new PDU and the UdpTxBuffer will be overwritten by the new PDU in case the transmit function returns with TCPIP_E_ARP_CACHE_MISS.

[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 `ForeRetrieve` set to `TRUE`.`()`

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

[SWS_SoAd_00543] The `Tcplp` module will retrieve the PDU data within the context of the `Tcplp` 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_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()`) 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()`) 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 an `nPduUdpTxBuffer` for this socket connection.`()`

[SWS_SoAd_00547] In case a `nPduUdpTxBuffer` is used for a socket connection and `TriggerMode` is set to `TRIGGER_NEVER` for the actual PDU (`SoAdPduRouteDest`), `SoAd` shall copy the PDU to the socket specific `nPduUdpTxBuffer` (instead of calling `TcpIp_UdpTransmit()`).`()`

[SWS_SoAd_00548] In case a `nPduUdpTxBuffer` 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 of the nPduUdpTxBuffer (if any) and the current PDU by calling `TcpIp_UdpTransmit()`.

[SWS_SoAd_00685] If In case a nPduUdpTxBuffer 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 of the nPduUdpTxBuffer (if any) by calling `TcpIp_UdpTransmit()` and then the current PDU by calling `TcpIp_UdpTransmit()` once more.

[SWS_SoAd_00549] If not enough space is available in the nPduUdpTxBuffer for the actual PDU with TriggerMode set to TRIGGER_NEVER, SoAd shall first transmit all PDUs of the nPduUdpTxBuffer by calling `TcpIp_UdpTransmit()` and then copy the PDU to the socket specific nPduUdpTxBuffer.

[SWS_SoAd_00690] SoAd shall preserve the order of PDUs transmitted via a socket connection that uses a nPduUdpTxBuffer (FIFO semantics). Pdu's collected on the sender side first shall be extracted and indicated to the receivers on the receiving side first as well.

[SWS_SoAd_00691] In case a nPduUdpTxBuffer is used for a socket connection SoAd shall store all PDUs individually, also PDUs with the same PduId.

[SWS_SoAd_00550] Within `SoAd_MainFunction()` SoAd shall transmit all PDUs of the nPduUdpTxBuffer by calling `TcpIp_UdpTransmit()` if nPduUdpTxBuffer is not empty and the specified `SoAdSocketUdpTriggerTimeout` expired.

[SWS_SoAd_00683] Socket Adapter shall start the nPduUdpTxBuffer specific timer, when configured by `SoAdSocketUdpTriggerTimeout(ECUC_SoAd_00133)`, when the first PDU is put into the buffer.

[SWS_SoAd_00684] Socket Adapter shall stop the nPduUdpTxBuffer specific timer, when configured by `SoAdSocketUdpTriggerTimeout(ECUC_SoAd_00133)`, when the buffer has been sent.

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_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.

⌋()

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 `SoAdSrcPduId` provided at `SoAd_TpTransmit()`.
- (3) Store the TP transmission request for further processing in the `SoAd_MainFunction()`.

⌋()

Note: `SoAdSrcPduId` 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 `ForeRetrieve` 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()`.₁()

[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.₁()

[SWS_SoAd_00668] 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_NOT_OK as result within the `SoAd_MainFunction()` after `TcpIp_UdpTransmit()` returns either with TCPIP_E_ARP_CACHE_MISS and `SoAdSocketUdpRetryEnabled` is set to FALSE or TCPIP_E_NOT_OK.₁()

[SWS_SoAd_00669] In case of a UDP socket connection the SoAd shall retry `TcpIp_UdpTransmit()` within the next `SoAd_MainFunction()` if `TcpIp_UdpTransmit()` returns with TCPIP_E_ARP_CACHE_MISS and `SoAdSocketUdpRetryEnabled` is set to TRUE.₁()

[SWS_SoAd_00667] In case of a TCP socket connection configured with `SoAdSocketTcplmmediateTpTxConfirmation` 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.₁()

[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.₁()

[SWS_SoAd_00558] In case of a TCP socket connection configured with `SoAdSocketTcplmmediateTpTxConfirmation` 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.₁()

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 (SoAdPduHeaderEnable 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 (SoAdPduHeader-Enable 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_00559] In case PDU header option is enabled (SoAdPduHeader-Enable 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 report SOAD_E_INV_PDUHEADER_ID to DET if DET is enabled.
- (4) use the length field of the PDU Header to identify the length of the actual PDU and the start of the next PDU

]()

[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]().

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.

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 `RxPduld` 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 `PdulInfoType.SduDataPtr` and `PdulInfoType.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 `RxPduld` 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 `RxPduld` as used at `<Up>_[SoAd][Tp]StartOfReception()` and `PdulInfoType.SduDataPtr` pointing to the PDU data provided by `SoAd_RxIndication()` and `PdulInfoType.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 remote 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 `SoAdSocketTcplInitiate`. `⌋()`

[SWS_SoAd_00587]⌈ SoAd shall return `E_NOT_OK` for TP-PDU Tx requests received at `SoAd_TpTransmit()` within connection reestablishment. `⌋()`

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 active or deactivate the state at initialization depending on the configuration parameter `SoAdIsEnabledAtInit`. `⌋()`

[SWS_SoAd_00560]⌈ If `SoAd_IfTransmit()` is called with `SoAdSrcPduId` 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 `SoAdSrcPduId` 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 receive 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.]()

Note: It is intended to make this behavior symmetrical and allow for more than one `SoAdSocketRouteDest` per `SoAdSocketRoute` for upper layers of If-type, i.e. one single PDU received from one socket connection can be forwarded as multiple IF-PDUs in the future.

[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()`) only once after transmission on all related socket connections succeeded.]()

7.11 Resource Management Option

[SWS_SoAd_00649] The Resource Management functionality shall be an optional functionality. I. e. SoAd implementations are AUTOSAR SoAd SWS compliant even if they do NOT implement the features described in the requirements of this chapter.] ()

[SWS_SoAd_00125] [When a PDU is in line to be transmitted the resource management will check if this socket is properly set up to do so. If so, it will transmit the data there and increment this sockets counter in the Socket Connection Table. If the counter should overflow all counter values are integer divided by 2.] ()

[SWS_SoAd_00126] [Should the required socket not be available, the resource management will check for the lowest counter in the Socket Connection Table and try to close this socket, making sure no data waiting to be transmitted is lost in the process. It will then establish the socket connection required for the current PDU to be transmitted and initiate transmission. The newly created socket will (at least implicitly) use the resources of the one closed in the process.] ()

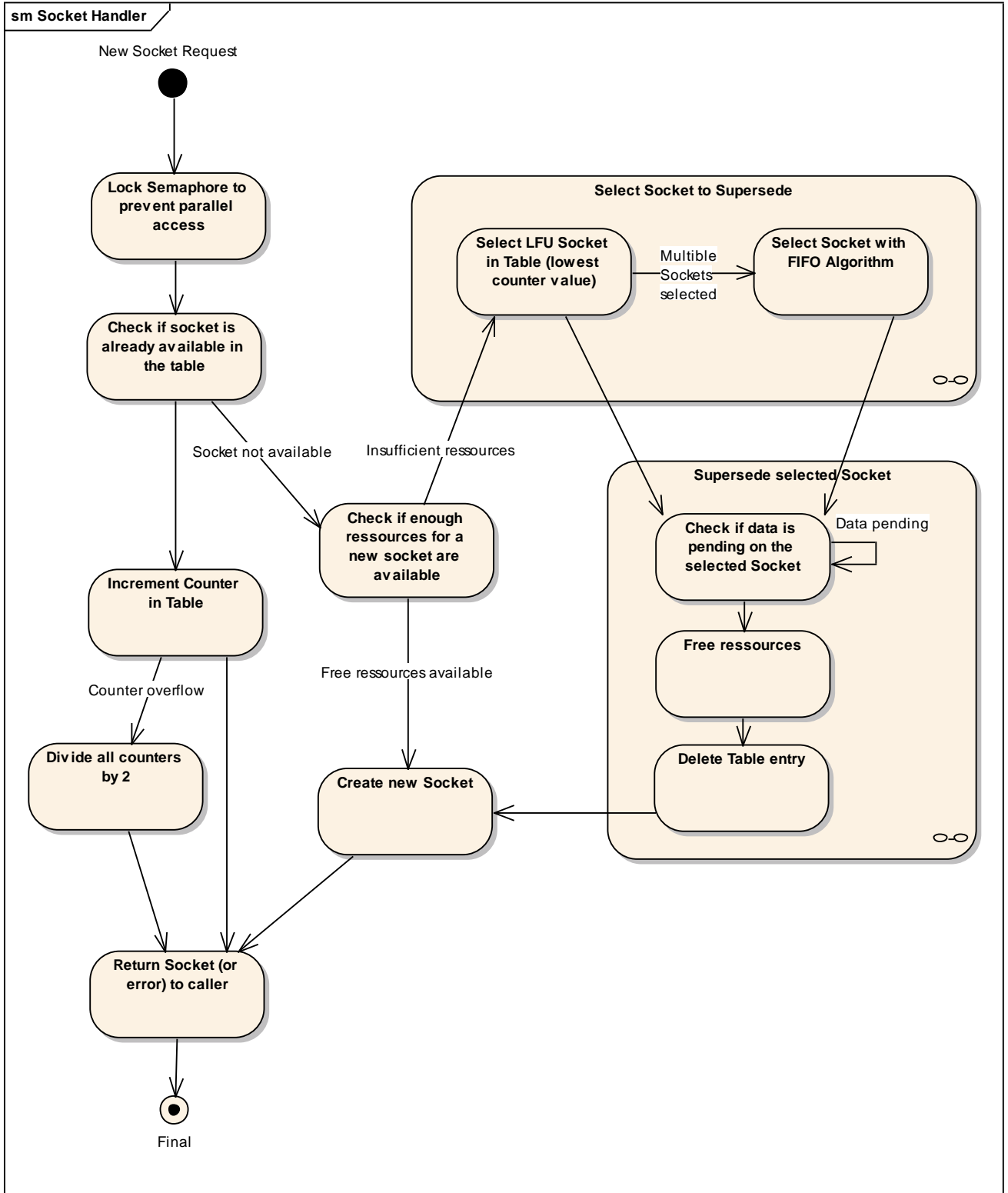


Figure 3: Socket Resource Management

7.12 Buffer handling

[SWS_SoAd_00505]「The SoAd shall provide sufficient buffers to store received data which can not 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 Tcplp.」()

[SWS_SoAd_00599]「 The SoAd shall provide sufficient buffers to store data which temporarily can not be forwarded to Tcplp, e.g. SoAd UDP TP transmit buffers or UdpTxBuffer, nPduUdpTxBuffer.」()

7.13 Error Handling

7.13.1 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.

The general requirements document of AUTOSAR [2] specifies that all basic software modules must distinguish (according to the product life cycle) two error types:

- **Development errors:** these errors should be detected and fixed during the development phase. In most cases, these errors are software errors. The detection errors that should only occur during development can be switched off for production code (by static configuration, namely preprocessor switches).
- **Production errors:** these errors are hardware errors and software exceptions that cannot be avoided and are expected to occur in the production (i.e. series) code. This kind of error is commonly known as a run-time error.

[SWS_SoAd_00101] 「The following table lists development error IDs the SoAd shall use for reporting of development errors to DET:

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value</i>
API service called before initializing the module	Development	SOAD_E_NOTINIT	0x01
API service called with NULL pointer	Development	SOAD_E_NULL_PTR	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

」()

Note: No production errors are reported by SoAd.

7.13.2 Error detection

For details refer to the chapter 7.3 “Error Detection” in *SWS_BSWGeneral*.

7.13.3 Error notification

For details refer to the chapter 7.4 “Error notification” in *SWS_BSWGeneral*.

7.14 Application notes

7.15 Debugging Concept

For details refer to the chapter 7.1.17 “Debugging support” in *SWS_BSWGeneral*.

7.16 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
Dem	Dem_EventIdType
	Dem_EventStatusType
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] ⌈

Name:	SoAd_SoConIdType
Type:	uint8, uint16
Description:	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]⌈

Name:	SoAd_SoConModeType		
Type:	Enumeration		
Range:	SOAD_SOCON_ONLINE	--	
	SOAD_SOCON_RECONNECT	--	
	SOAD_SOCON_OFFLINE	--	
Description:	type to specify the state of a SoAd socket connection.		

⌋()

8.2.3 SoAd_RoutingGroupIdType

[SWS_SoAd_00519]⌈

Name:	SoAd_RoutingGroupIdType		
Type:	uint8, uint16		
Description:	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]⌈

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

⌋()

8.3 Function definitions

8.3.1 General

8.3.1.1 SoAd_GetVersionInfo

[SWS_SoAd_00096] ⌈

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

⌋()

8.3.1.2 SoAd_Init

[SWS_SoAd_00093] [

Service name:	SoAd_Init	
Syntax:	<pre>void SoAd_Init(const SoAd_ConfigType* SoAdConfigPtr)</pre>	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SoAdConfigPtr	Pointer to the configuration data of the SoAd module.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	<p>Description: This service initializes all global variables of a Socket Adaptor instance and puts it into the idle state. It has no return value because software errors in initialization data shall be detected during configuration time (e.g. by configuration tool). Furthermore, if a hardware error occurs it shall be reported via the error manager modules.</p> <p>Caveats: The call of this service is mandatory before using the SoAd instance for further processing. The API has to be called during initialization.</p>	

] ()

[SWS_SoAd_00211] [

SoAd_Init shall store the access to the configuration structure for subsequent API calls.] ()

[SWS_SoAd_00212] [

Configuration: The user shall pass the post-build configuration or a NULL_PTR as parameter depending on the configuration variant.] ()

[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_INV_ARG.] ()

8.3.2 Normal Operation

8.3.2.1 SoAd_IfTransmit

[SWS_SoAd_00091] [

Service name:	SoAd_IfTransmit	
Syntax:	Std_ReturnType SoAd_IfTransmit(PduIdType SoAdSrcPduId, const PduInfoType* SoAdSrcPduInfoPtr)	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	SoAdSrcPduId	This parameter contains a unique identifier referencing to the PDU Routing Table and thereby specifying the socket to be used for transmission of the data.
	SoAdSrcPduInfoPtr	A pointer to a structure with socket related data: data length and pointer to a data buffer.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted, e.g. due to a still ongoing transmission in the corresponding socket or the to be transmitted message is too long.
Description:	<p>This service initiates a request for transmission of the L-PDU specified by the SoAdSrcPduId. The corresponding socket has to be resolved by the SoAdSrcPduId.</p> <p>This call is used to mimic the call to an IF in AUTOSAR.</p> <p>Development errors: Invalid values of SoAdSrcPduId or SoAdSrcPduInfoPtr will be reported to the development error tracer (SOAD_E_INVALID_TXPDUID or SOAD_E_PARAM_POINTER).</p>	

] ()

[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 and return E_NOT_OK.] ()

[SWS_SoAd_00214] [

If development error detection is enabled: SoAd_IfTransmit() shall check parameter SoAdSrcPduId for being valid. If the check fails, SoAd_IfTransmit() shall raise the development error SOAD_E_INV_PDUID and return E_NOT_OK.] ()

[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 SoAdSrcPduId.] ()

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] ⌈

Service name:	SoAd>IfRoutingGroupTransmit	
Syntax:	Std_ReturnType SoAd>IfRoutingGroupTransmit (SoAd_RoutingGroupIdType id)	
Service ID[hex]:	0x1D	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	routing group identifier indirectly specifying PDUs to be transmitted (after requesting the newest data from the related upper layer).
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
Description:	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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

8.3.2.3 SoAd_TpTransmit

[SWS_SoAd_00105] [

Service name:	SoAd_TpTransmit	
Syntax:	Std_ReturnType SoAd_TpTransmit(PduIdType SoAdSrcPduId, const PduInfoType* SoAdSrcPduInfoPtr)	
Service ID[hex]:	0x04	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	SoAdSrcPduId	This parameter contains a unique identifier referencing to the PDU Routing Table and thereby specifying the socket to be used for transmission of the data.
	SoAdSrcPduInfoPtr	A pointer to a structure with socket related data. Only the length data is valid.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted, e.g. due to a still ongoing transmission in the corresponding socket or the to be transmitted message is too long.
Description:	<p>This service is utilized to request the transfer of data. It sets a flag for indicating that a transmit request is present.</p> <p>This function has to be called with the PDU-ID of the SoAd, i.e. the upper layer has to translate its own PDU-ID into the one of the SoAd for the corresponding message.</p> <p>This call shall fail if no entry for the PDU-ID can be found in the socket connection table.</p> <p>Within the provided SoAdSrcPduInfoPtr only SduLength is valid (no data)! If this function returns E_OK then there will arise a call of <Up>_[SoAd][Tp]CopyTxData in order to get data for sending.</p> <p>This call is used to mimic the call to a TP in AUTOSAR.</p>	

] ()

[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 and return E_NOT_OK.] ()

[SWS_SoAd_00237] [

If development error detection is enabled: SoAd_TpTransmit() shall check parameter SoAdSrcPduId for being valid. If the check fails, SoAd_TpTransmit() shall raise the development error SOAD_E_INV_PDUID and return E_NOT_OK.] ()

[SWS_SoAd_00650] If 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 `SoAdSrcPduId`.)()

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] If

Service name:	SoAd_TpCancelTransmit	
Syntax:	Std_ReturnType SoAd_TpCancelTransmit (PduIdType PduId)	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	PduId	Identifiacion of the I-PDU to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation: E_OK: request accepted (but not yet performed). E_NOT_OK: request not accepted (e.g. cancellation not possible).
Description:	Requests cancellation of the transmission via TP for a specific I-PDU.	

)()

[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` and return `E_NOT_OK`.)()

[SWS_SoAd_00606] If development error detection is enabled: `SoAd_TpCancelTransmit()` shall check parameter `PduId` for being valid. If the check fails, `SoAd_TpCancelTransmit()` shall raise the development error `SOAD_E_INV_PDUID` and return `E_NOT_OK`.)()

8.3.3.2 SoAd_TpCancelReceive

[SWS_SoAd_00521]⌈

Service name:	SoAd_TpCancelReceive	
Syntax:	Std_ReturnType SoAd_TpCancelReceive(PduIdType PduId)	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	PduId	Identification of the I-PDU to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation: E_OK: request accepted (but not yet performed). E_NOT_OK: request not accepted (e.g. cancellation not possible).
Description:	Requests cancellation of the reception via TP for a specific I-PDU.	

⌋()

[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 and return E_NOT_OK.⌋()

[SWS_SoAd_00608]⌈ If development error detection is enabled: SoAd_TpCancelReceive() shall check parameter PduId for being valid. If the check fails, SoAd_TpCancelReceive() shall raise the development error SOAD_E_INV_PDUID and return E_NOT_OK.⌋()

8.3.4 Information and Control API

8.3.4.1 SoAd_GetSoConId

[SWS_SoAd_00509]⌈

Service name:	SoAd_GetSoConId	
Syntax:	Std_ReturnType SoAd_GetSoConId(PduIdType TxPduId, SoAd_SoConIdType* SoConIdPtr)	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	TxPduId	Transmit PduId specifying the SoAd socket connection for which the socket connection index shall be returned.
Parameters	None	

(inout):		
Parameters (out):	SoConIdPtr	Pointer to memory receiving the socket connection index asked for.
Return value:	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful
Description:	Returns socket connection index related to specified transmit PduId. In case a fan-out is configured for TxPduId (i.e. multiple SoAdPduRouteDest specified) E_NOT_OK shall be returned.	

]()

[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 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 and return E_NOT_OK.]()

8.3.4.2 SoAd_OpenSoCon

[SWS_SoAd_00510]

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

[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` and return `E_NOT_OK.()`

[SWS_SoAd_00528] 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 reported and `SoAd_OpenSoCon()` shall return `E_NOT_OK.()`

8.3.4.3 SoAd_CloseSoCon

[SWS_SoAd_00511]

Service name:	SoAd_CloseSoCon	
Syntax:	Std_ReturnType SoAd_CloseSoCon(SoAd_SoConIdType SoConId, boolean abort)	
Service ID[hex]:	0x09	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
Description:	This service closes the socket connection specified by <code>SoConId</code> .	

.)()

[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` and return `E_NOT_OK.()`

[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` and return `E_NOT_OK.()`

[SWS_SoAd_00529] 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 reported and SoAd_CloseSoCon () shall return E_NOT_OK.」()

8.3.4.4 SoAd_RequestIpAddrAssignment

[SWS_SoAd_00520]「

Service name:	SoAd_RequestIpAddrAssignment	
Syntax:	Std_ReturnType SoAd_RequestIpAddrAssignment (SoAd_SoConIdType SoConId, TcpIp_IpAddrAssignmentType Type, TcpIp_SockAddrType* LocalIpAddrPtr)	
Service ID[hex]:	0x0A	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
Parameters (in):	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 use in case paramter Type is set to TCPIP_IPADDR_ASSIGNMENT_STATIC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description:	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 and return E_NOT_OK.」()

[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 and return E_NOT_OK.」()

8.3.4.5 SoAd_ReleaseIpAddrAssignment

[SWS_SoAd_00536]⌈

Service name:	SoAd_ReleaseIpAddrAssignment	
Syntax:	Std_ReturnType SoAd_ReleaseIpAddrAssignment (SoAd_SoConIdType SoConId)	
Service ID[hex]:	0x0B	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	SoConId	socket connection index specifying the socket connection for which the IP address shall be released
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description:	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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

8.3.4.6 SoAd_GetLocalAddr

[SWS_SoAd_00506]⌈

Service name:	SoAd_GetLocalAddr	
Syntax:	<pre>Std_ReturnType SoAd_GetLocalAddr(SoAd_SoConIdType SoConId, TcpIp_SockAddrType* LocalAddrPtr, uint8* NetmaskPtr, TcpIp_SockAddrType* DefaultRouterPtr)</pre>	
Service ID[hex]:	0x0C	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	SoConId	socket connection index representing the SoAd socket connection for which the actual local IP address shall be obtained.
Parameters (inout):	None	
Parameters (out):	LocalAddrPtr	Pointer to a struct where the local address (IP address and port) is stored.
	NetmaskPtr	Pointer to memory where Network mask of IPv4 address or address prefix of IPv6 address in CIDR Notation is stored
	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)
Return value:	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
Description:	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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

8.3.4.7 SoAd_GetPhysAddr

[SWS_SoAd_00507]⌈

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

[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 and return E_NOT_OK.⌋()

8.3.4.8 SoAd_GetRemoteAddr

[SWS_SoAd_00655] ⌈

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

⌋()

[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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

[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.⌋()

8.3.4.9 SoAd_EnableRouting

[SWS_SoAd_00516]⌈

Service name:	SoAd_EnableRouting	
Syntax:	Std_ReturnType SoAd_EnableRouting(SoAd_RoutingGroupIdType id)	
Service ID[hex]:	0x0E	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	routing group identifier specifying the routing group to be enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
Description:	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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

8.3.4.10 SoAd_DisableRouting

[SWS_SoAd_00517]⌈

Service name:	SoAd_DisableRouting	
Syntax:	Std_ReturnType SoAd_DisableRouting(SoAd_RoutingGroupIdType id)	
Service ID[hex]:	0x0F	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	routing group identifier specifying the routing group to be disabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation E_OK The request was successful E_NOT_OK The request was not successful.
Description:	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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

8.3.4.11 SoAd_SetRemoteAddr

[SWS_SoAd_00515]⌈

Service name:	SoAd_SetRemoteAddr	
Syntax:	Std_ReturnType SoAd_SetRemoteAddr(SoAd_SoConIdType SoConId, TcpIp_SockAddrType* RemoteAddrPtr)	
Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
Parameters (in):	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.

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
Description:	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 and return E_NOT_OK.]()

[SWS_SoAd_00531] If DET is enabled and SoConId refers to a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to TRUE, the function SoAd_SetRemoteAddr() shall return with E_NOT_OK and report SOAD_E_INV_ARG to DET.]()

[SWS_SoAd_00532] The function SoAd_SetRemoteAddr() shall only proceed if SoConId refers to

- (3) a socket connection that is in the mode SOAD_SOCON_OFFLINE or
- (4) a UDP socket connection that has no active TP session (i.e. no reception or transmission via the TP-API ongoing).

Otherwise the request shall be rejected and return with E_NOT_OK.]()

[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.]()

8.3.4.12 SoAd_SetUniqueRemoteAddr

[SWS_SoAd_00671]

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

⌋()

[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 and return E_NOT_OK.⌋()

[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 and return E_NOT_OK.⌋()

[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] If 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.]()

8.3.4.13 SoAd_TpChangeParameter

[SWS_SoAd_00508] If

Service name:	SoAd_TpChangeParameter	
Syntax:	Std_ReturnType SoAd_TpChangeParameter(PduIdType id, TPParameterType parameter, uint16 value)	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
Parameters (in):	id	Identification of the I-PDU which the parameter change shall affect
	parameter	Identifier of the parameter to be changed
	value	New parameter value
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted
	Description: By this API service the SoAd or TCP/IP stack is requested to change a connection parameter. E.g. the Nagle algorithm may be controlled by this API.	

]()

[SWS_SoAd_00629] If development error detection is enabled: SoAd_TpChangeParameter() shall check that the service SoAd_Init was previously called. If the check fails, SoAd_TpChangeParameter() shall raise the development error SOAD_E_NOTINIT and return E_NOT_OK.]()

[SWS_SoAd_00631] If development error detection is enabled: SoAd_TpChangeParameter() shall check parameter id for being valid. If the check fails, SoAd_TpChangeParameter() shall raise the development error SOAD_E_INV_ARG and return E_NOT_OK.]()

[SWS_SoAd_00630] If development error detection is enabled: SoAd_TpChangeParameter() shall check parameter parameter for being valid. If

the check fails, `SoAd_TpChangeParameter()` shall raise the development error `SOAD_E_INV_ARG` and return `E_NOT_OK.()`

8.3.4.14 SoAd_ReadDhcpHostNameOption

[SWS_SoAd_00681] ⌈

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

⌋()

8.3.4.15 SoAd_WriteDhcpHostNameOption

[SWS_SoAd_00679] ⌈

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

⌋()

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] [

Service name:	SoAd_RxIndication	
Syntax:	<pre>void SoAd_RxIndication(TcpIp_SocketIdType SocketId, TcpIp_SockAddrType* RemoteAddrPtr, uint8* BufPtr, uint16 Length)</pre>	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SocketIds. Non reentrant for the same SocketId.	
Parameters (in):	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	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 and return E_NOT_OK.] ()

[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` and return `E_NOT_OK`. ⌋()

8.4.2 SoAd_CopyTxData

[SWS_SoAd_00523] ⌈

Service name:	SoAd_CopyTxData	
Syntax:	<pre>BufReq_ReturnType SoAd_CopyTxData(TcpIp_SocketIdType SocketId, uint8* BufPtr, uint16 BufLength)</pre>	
Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SocketIds. Non reentrant for the same SocketId.	
Parameters (in):	SocketId	Socket identifier of the related local socket resource.
	BufPtr	Pointer to buffer for transmission data.
	BufLength	Length of provided data buffer.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	BufReq_ReturnType	<p>BUFREQ_OK: Data has been copied to the transmit buffer completely as requested.</p> <p>BUFREQ_E_NOT_OK: Data has not been copied. Request failed. (No further action for Tcplp required. Later SoAd might either close the socket or retry the transmit request)</p>
Description:	<p>This service requests to copy data for transmission to the buffer indicated. This call is triggered by <code>Tcplp_Transmit()</code>. Note: The call to <code>SoAd_CopyTxData()</code> may happen in the context of <code>Tcplp_Transmit()</code>.</p>	

⌋()

[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` and return `E_NOT_OK`. ⌋()

[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` and return `E_NOT_OK`. ⌋()

8.4.3 SoAd_TxConfirmation

[SWS_SoAd_00098] [

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

] ()

[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 and return E_NOT_OK.] ()

[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 and return E_NOT_OK.] ()

8.4.4 SoAd_TcpAccepted

[SWS_SoAd_00099] [

Service name:	SoAd_TcpAccepted	
Syntax:	Std_ReturnType SoAd_TcpAccepted(TcpIp_SocketIdType SocketId, TcpIp_SocketIdType SocketIdConnected, TcpIp_SockAddrType* RemoteAddrPtr)	
Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	SocketId	Socket identifier of the related local socket resource which has been used at Tcplp_Bind()
	SocketIdConnected	Socket identifier of the local socket resource used for the established connection.
	RemoteAddrPtr	IP address and port of the remote host.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	Result of operation E_OK SoAd accepts the established connection E_NOT_OK SoAd refuses the established connection, Tcplp stack shall close the connection.
Description:	SoAd_TcpAccepted() 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 Tcplp_TcpListen() 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 and return E_NOT_OK.] ()

8.4.5 SoAd_TcpConnected

[SWS_SoAd_00100] [

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

8.4.6 SoAd_TcplpEvent

[SWS_SoAd_00146] [

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

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	SoAd_TcpIpEvent() gets called if the stack encounters a condition described by the values in TcpIpEvent.

] ()

[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 and return E_NOT_OK.] ()

[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 and return E_NOT_OK.] ()

8.4.7 SoAd_LocalIpAddrAssignmentChg

[SWS_SoAd_00209] [

Service name:	SoAd_LocalIpAddrAssignmentChg	
Syntax:	void SoAd_LocalIpAddrAssignmentChg (TcpIp_LocalAddrIdType IpAddrId, TcpIp_IpAddrStateType State)	
Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	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
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	SoAd_LocalIpAssignmentChg() 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_LocalIpAddressAssignmentChg() shall raise the development error SOAD_E_NOTINIT.] ()

[SWS_SoAd_00280] [

If development error detection is enabled: SoAd_LocalIpAddressAssignmentChg() shall check parameter IpAddrId for being valid. If the check fails, SoAd_LocalIpAddressAssignmentChg() shall raise the development error SOAD_E_INV_ARG and return E_NOT_OK.] ()

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] [

Service name:	SoAd_MainFunction
Syntax:	void SoAd_MainFunction(void)
Service ID[hex]:	0x19
Description:	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() is 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
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
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_GetSocket	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_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_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.

└()

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]┌

Service name:	<Up>_[SoAd][If]RxIndication	
Syntax:	void <Up>_[SoAd][If]RxIndication(PduIdType RxPduId, const PduInfoType* PduInfoPtr)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	ID of the received I-PDU
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Indication of a received I-PDU from a lower layer communication module.

] ()

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

[SWS_SoAd_00663]

Service name:	<Up>_[SoAd][If]TriggerTransmit	
Syntax:	Std_ReturnType <Up>_[SoAd][If]TriggerTransmit(PduIdType TxPduId, PduInfoType* PduInfoPtr)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

]()

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

[SWS_SoAd_00107]

Service name:	<Up>_[SoAd][If]TxConfirmation	
Syntax:	void <Up>_[SoAd][If]TxConfirmation(PduIdType TxPduId)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the I-PDU that has been transmitted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication module confirms the transmission of an I-PDU.	

] ()

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

[SWS_SoAd_00138] [

Service name:	<Up>_[SoAd][Tp]StartOfReception	
Syntax:	BufReq_ReturnType <Up>_[SoAd][Tp]StartOfReception(PduIdType RxPduId, const PduInfoType* info, PduLengthType TpSduLength, PduLengthType* bufferSizePtr)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different Pdulds. Non reentrant for the same Pdul.	
Parameters (in):	RxPdul	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. Depending on the global parameter MetaDataLength, additional bytes containing MetaData (e.g. the CAN ID) are appended after the payload data.
	TpSduLength	Total length of the PDU to be received.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute Block Size (BS) in the transport protocol module.
Return value:	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted. BufferSizePtr indicates the available receive buffer. BUFREQ_E_NOT_OK: Connection has been rejected. RxBufferSizePtr remains unchanged. BUFREQ_E_OVFL: No Buffer of the required length can be provided.
Description:	This function will be called by the transport protocol module at the start of receiving an I-PDU. The I-PDU 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 also when invoked with TpSdulength equal to 0.	

] ()

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

[SWS_SoAd_00139] [

Service name:	<Up>_[SoAd][Tp]CopyRxData	
Syntax:	BufReq_ReturnType <Up>_[SoAd][Tp]CopyRxData(PduIdType RxPduId, const PduInfoType* PduInfoPtr, PduLengthType* bufferSizePtr)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different Pdulds. Non reentrant for the same Pdul.	
Parameters (in):	RxPdul	Identification of the received I-PDU.
	PduInfoPtr	Provides a pointer to the data and the number of bytes to be copied by the upper layer. A size of 0 can be used to by the SoAd to query the available buffer size in the upper layer

		module.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer after data has been copied.
Return value:	BufReq_ReturnType	BUFREQ_OK: Buffer request accomplished successful. BUFREQ_E_NOT_OK: Buffer request not successful. Buffer cannot be accessed.
Description:	This function is called when transport protocol module have data to copy to the receiving module. Several calls may be made during one transportation of an I-PDU.	

] ()

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

[SWS_SoAd_00180] [

Service name:	<Up>_[SoAd][Tp]RxIndication	
Syntax:	void <Up>_[SoAd][Tp]RxIndication(PduIdType RxPduId, Std_ReturnType result)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	Identification of the received I-PDU.
	result	Result of the reception
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Called by the transport protocol module after an I-PDU has been received successfully or when an error occurred.	

] ()

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

[SWS_SoAd_00137] [

Service name:	<Up>_[SoAd][Tp]CopyTxData	
Syntax:	BufReq_ReturnType <Up>_[SoAd][Tp]CopyTxData(PduIdType TxPduId, const PduInfoType* PduInfoPtr, RetryInfoType* retry, PduLengthType* availableDataPtr)	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TxPduId	Identification of the transmitted I-PDU.
	PduInfoPtr	Provides destination buffer and the number of bytes to copy. A size of copy size of 0 can be used by the SoAd to query the available buffer size in the upper layer module.
	retry	Parameter is always set to NULL_PTR as SoAd does not support re-transmit. Re-transmission on TCP level is backed with data already buffered in TcpIp, so there is no need to re-

		query the upper layer.
Parameters (inout):	None	
Parameters (out):	availableDataPtr	Indicates the remaining number of bytes that are available in the UL Tx buffer.
Return value:	BufReq_ReturnType	<p>BUFREQ_OK: Data has been copied to the transmit buffer completely as requested.</p> <p>BUFREQ_E_BUSY: Request could not be fulfilled as the required amount of Tx data is not available. TP layer might retry later on. No data has been copied.</p> <p>BUFREQ_E_NOT_OK: Data has not been copied. Request failed.</p>
Description:	This function is called by the SoAd to query the transmit data of an I-PDU segment. Each call to this function copies the next part of the transmit data until the entire data has been copied or the UL returned BUFREQ_E_NOT_OK.	

] ()

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

[SWS_SoAd_00181] [

Service name:	<Up>_[SoAd][Tp]TxConfirmation	
Syntax:	<pre>void <Up>_[SoAd][Tp]TxConfirmation(PduIdType TxPduId, Std_ReturnType result)</pre>	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TxPduld	Identification of the transmitted I-PDU.
	result	Result of the transmission of the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by a transport protocol module after the I-PDU has been transmitted on its network, the result will reveal if the transmission was successful or not.	

] ()

8.6.3.9 <Up>_SoConModeChg

[SWS_SoAd_00514]

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

⌋()

8.6.3.10 <Up>_LocalIpAddrAssignmentChg

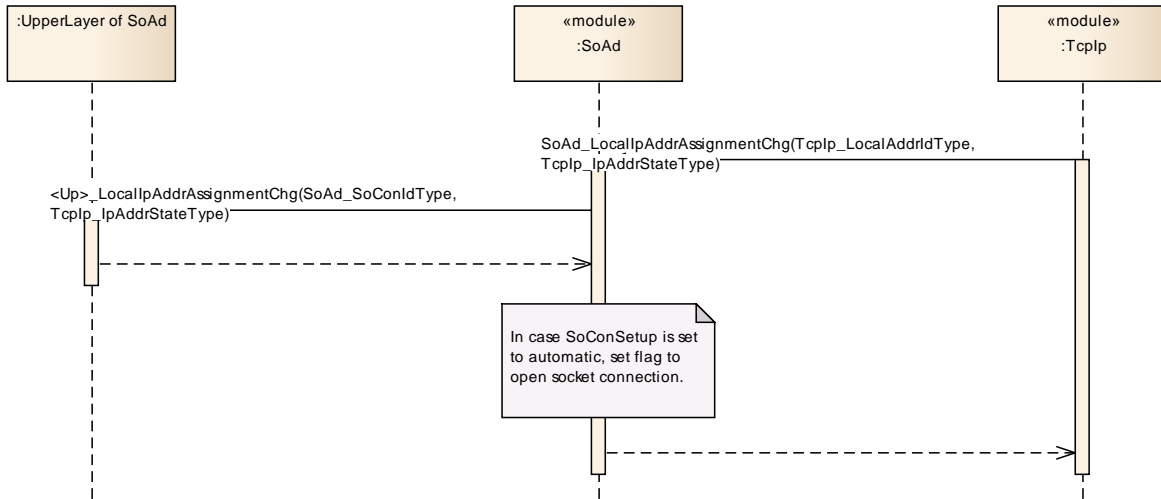
[SWS_SoAd_00513]

Service name:	<Up>_LocalIpAddrAssignmentChg	
Syntax:	<pre>void <Up>_LocalIpAddrAssignmentChg (SoAd_SoConIdType SoConId, TcpIp_IpAddrStateType State)</pre>	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
Parameters (in):	SoConId	socket connection index specifying the socket connection where the IP address assignment has changed
	State	state of IP address assignment
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	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).	

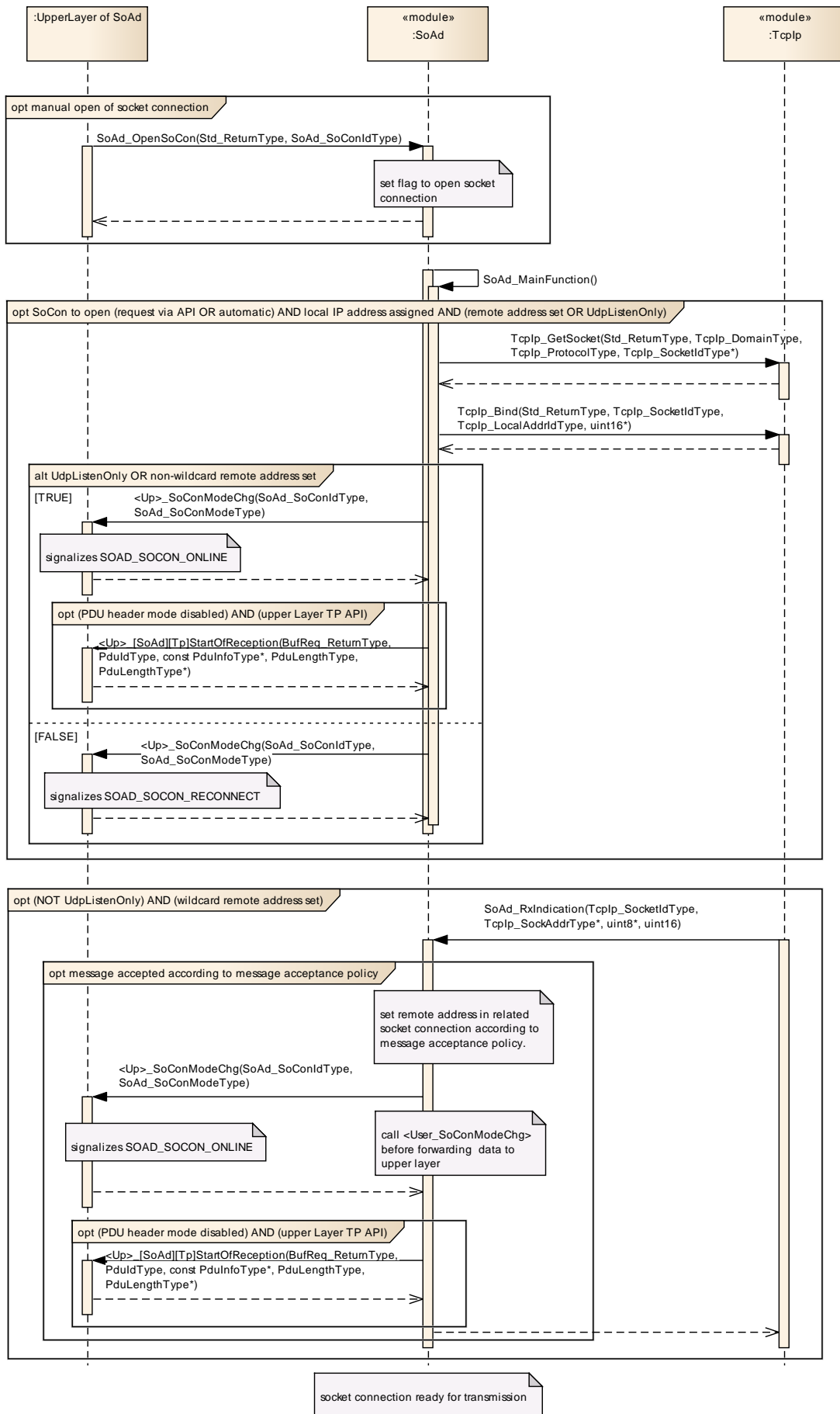
⌋()

9 Sequence diagrams and Transition Tables

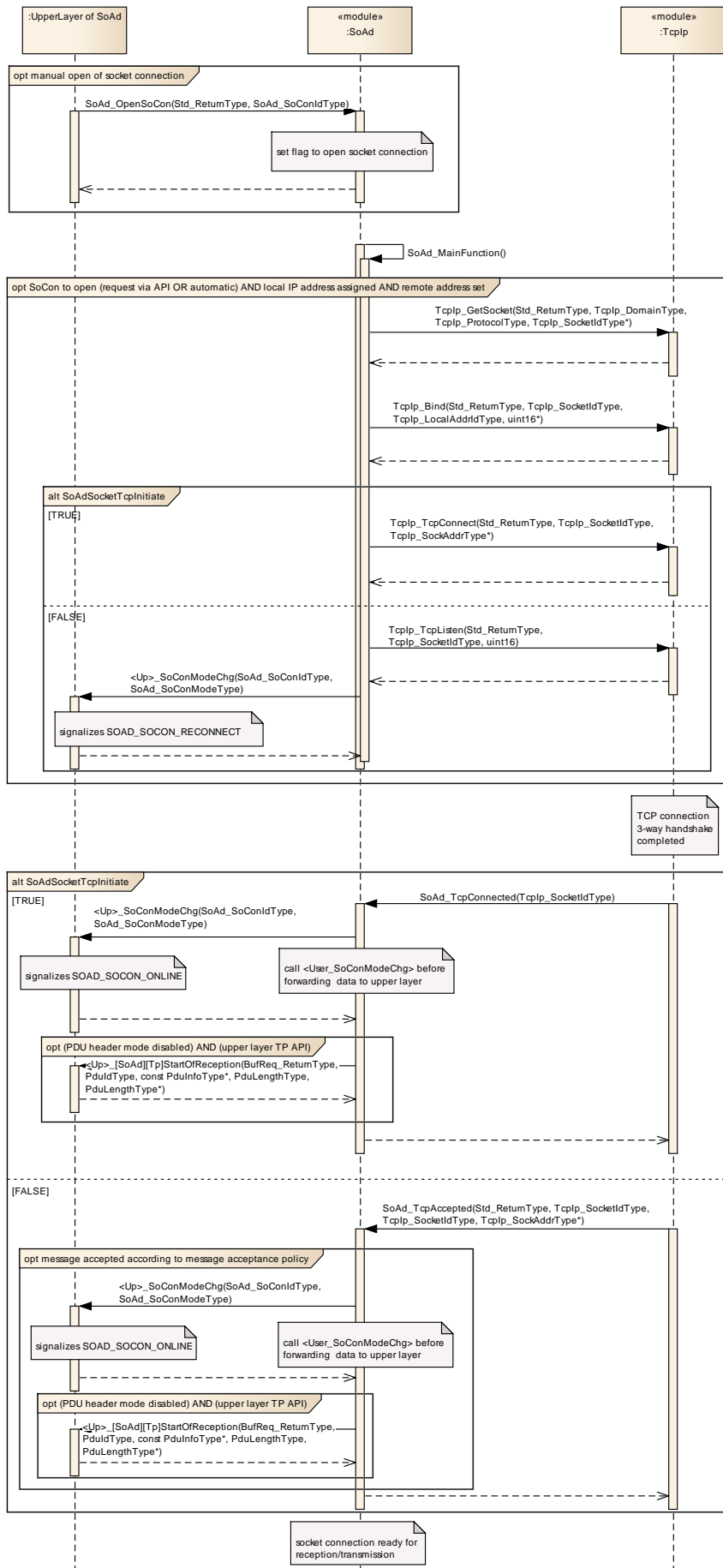
9.1 Address – Assignment



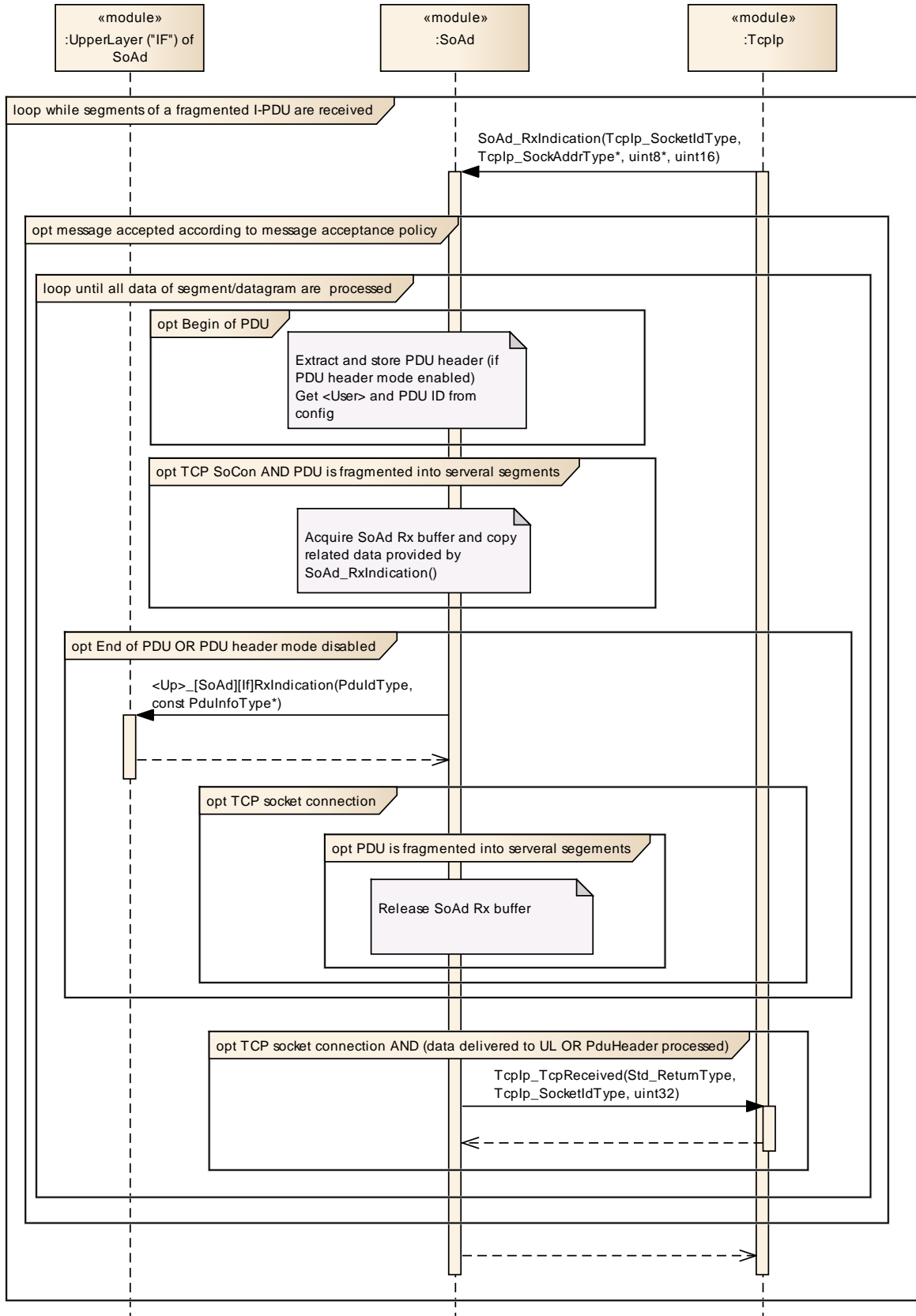
9.2 Socket Connection Setup – UDP



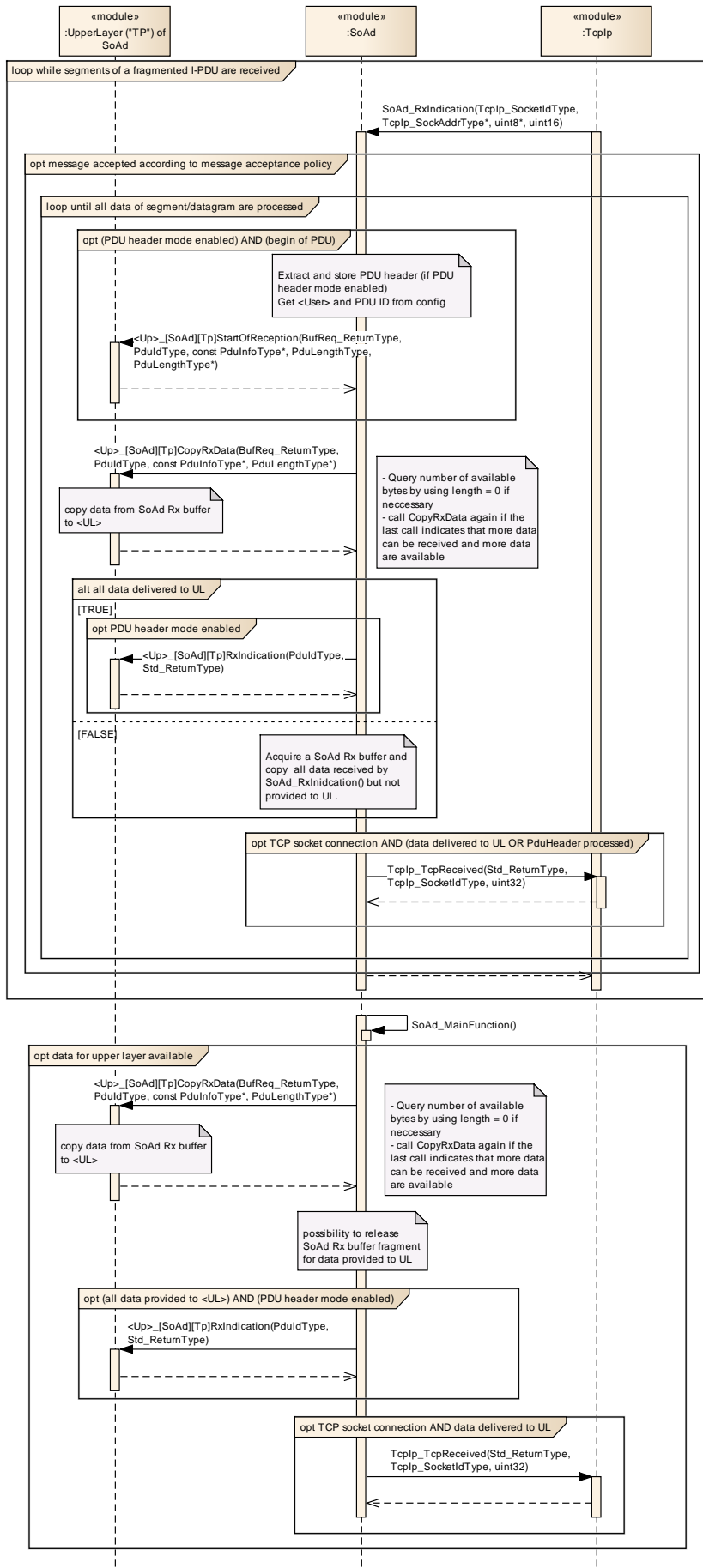
9.3 Socket Connection Setup – TCP



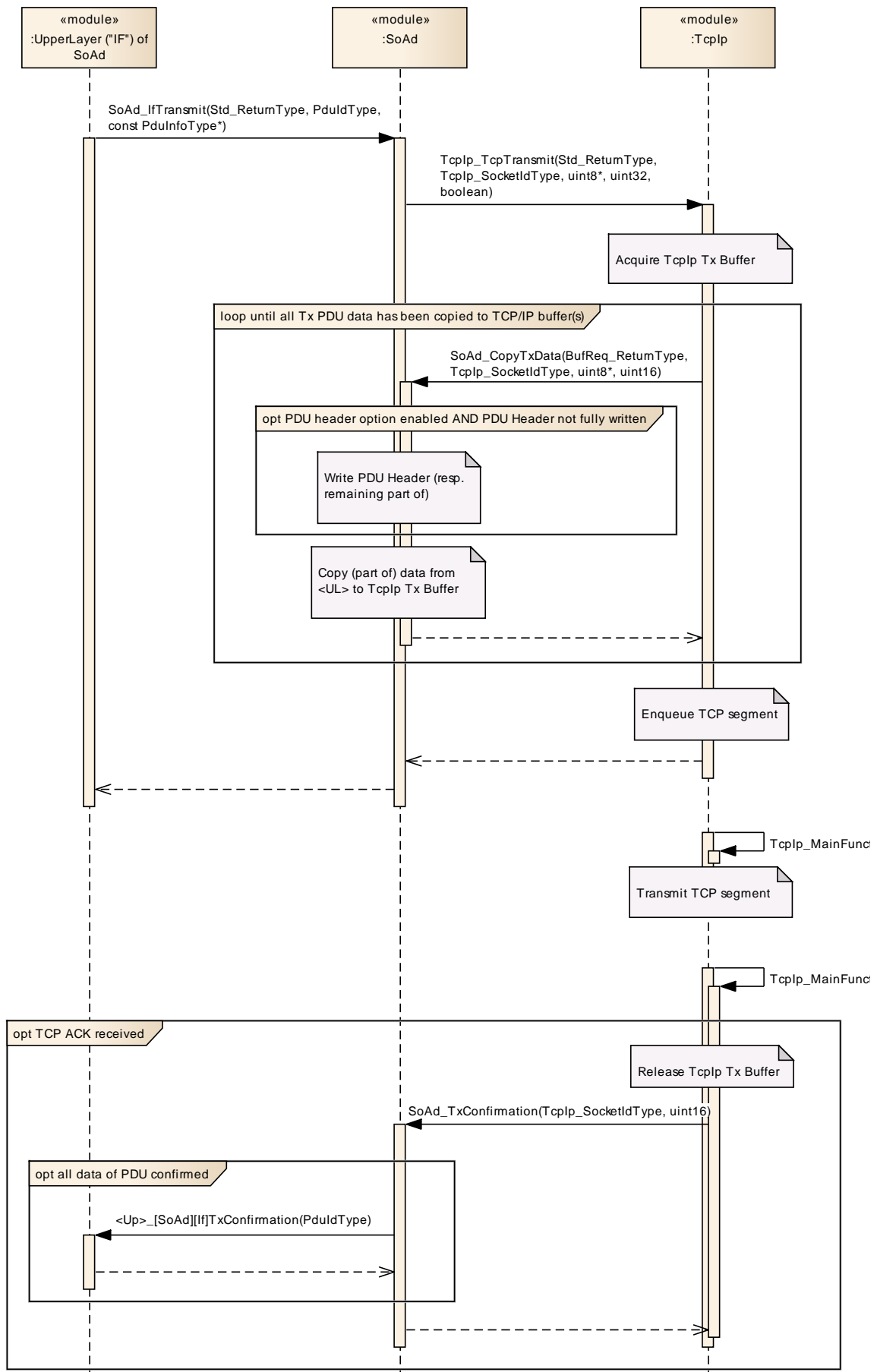
9.4 Reception – Upper Layer If API



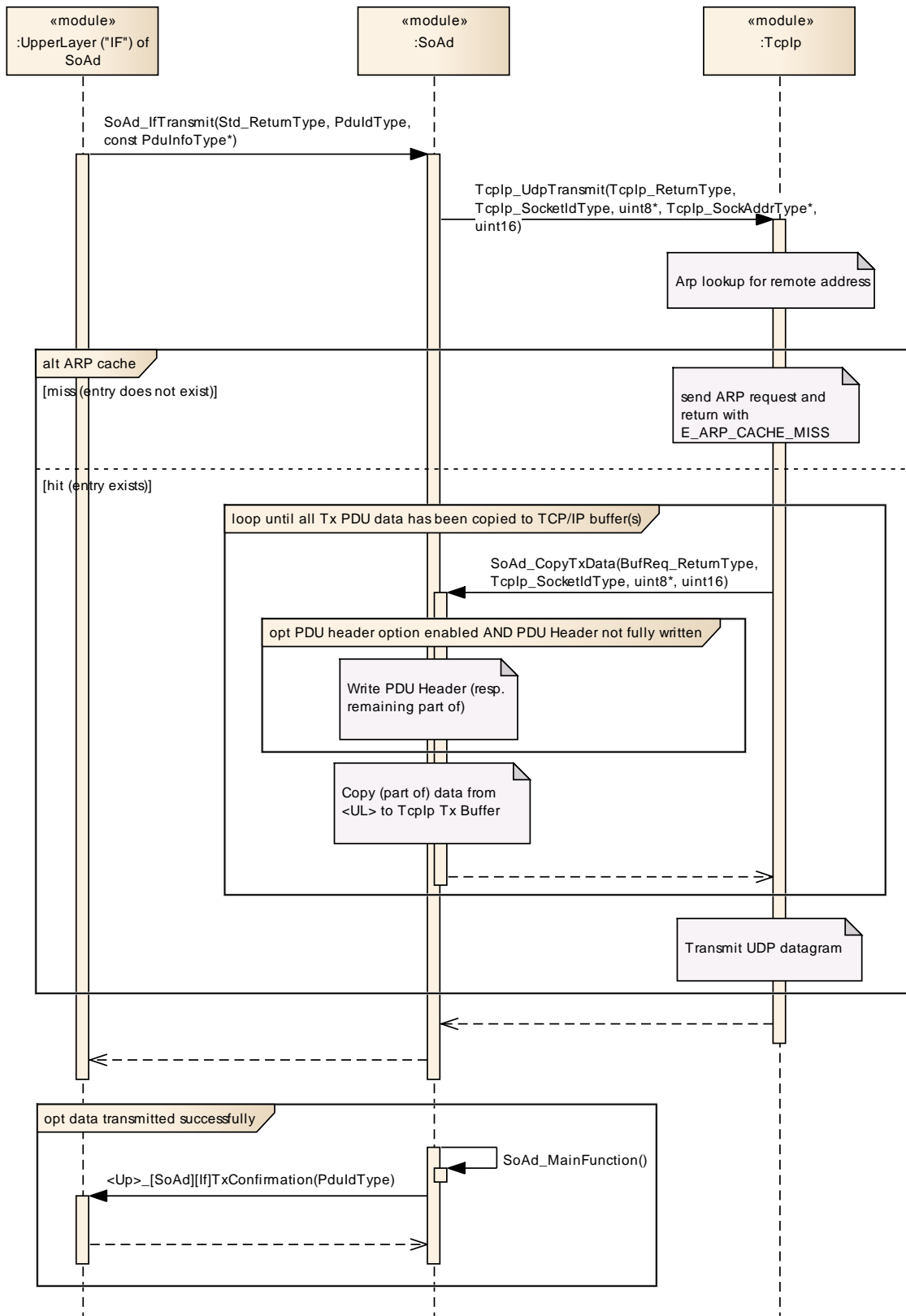
9.5 Reception – Upper Layer TP API



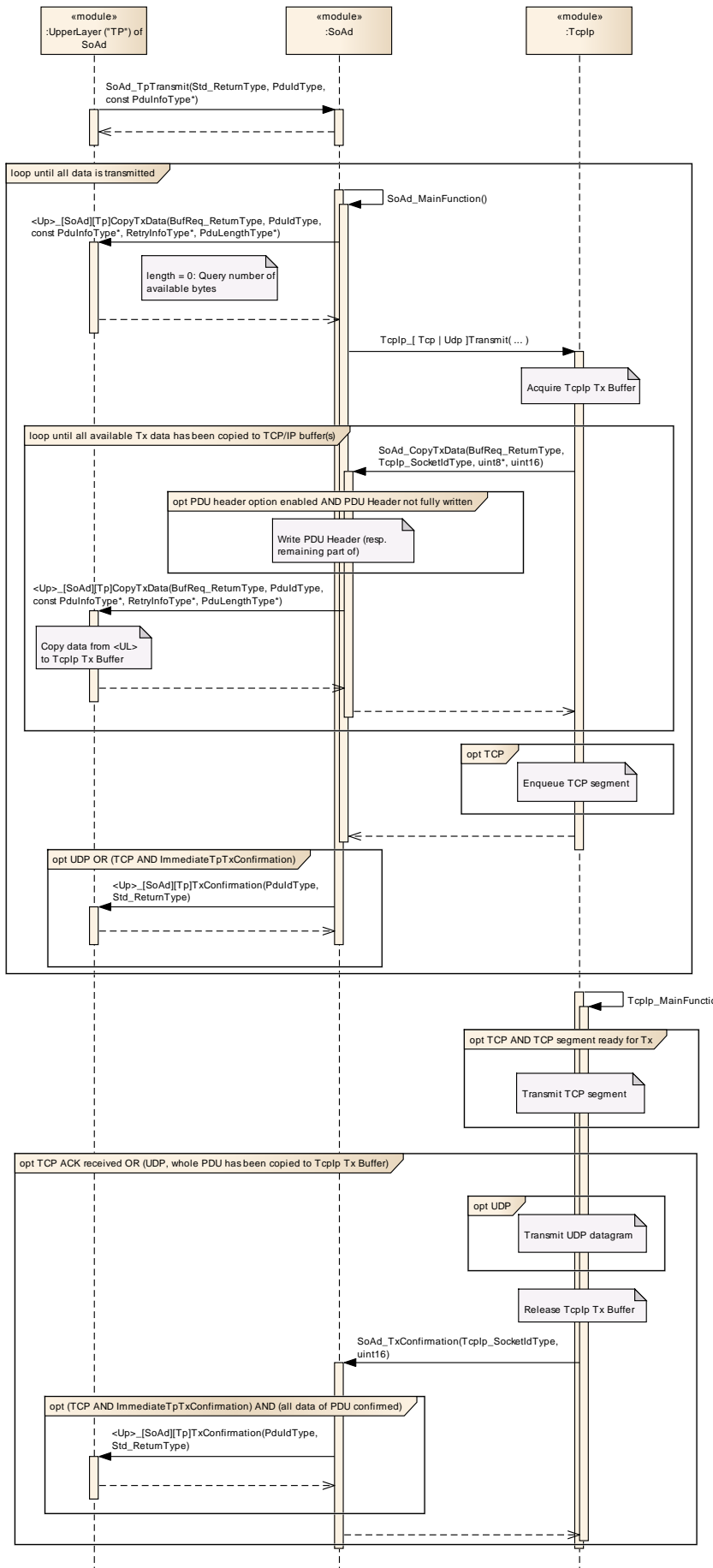
9.6 Transmission – Upper Layer If API – TCP



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 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 not necessarily be implemented as FLOAT, but their type and unit shall be adapted to the values required.] ()

10.2.1 Variants

Variant 1: All configuration parameters shall be configurable at pre-compile time.
Use case: Source code optimization.

Variant 2: All configuration parameters of the container `SoAd_GlobalConfig` related to enable or disable an optional feature shall be configurable at pre-compile time; the remaining configuration parameters shall be configurable at link time.

Use case: Object code.

Variant 3: The parameters contained in `SoAd_ChannelConfig` are configurable at post-build time. The parameters contained in `SoAd_GlobalConfig` are configurable at pre-compile time

Use case: ECU configuration can be flashed (L) and selected during initialization phase (M).

Note: The possibility to select a configuration (post-build time type L) is explicitly mentioned for Variant 3 only, but from a technical perspective it is also possible to provide this configuration variant for variant 1 and 2.

10.2.2 SoAd

SWS Item	ECUC_SoAd_00001 :
Module Name	SoAd
Module Description	Configuration of the SoAd (Socket Adaptor) module.

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
SoAdGeneral	1	This container contains all global configuration parameters of SoAd.

10.2.3 SoAdBswModules

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

SWS Item	ECUC_SoAd_00104 :		
Name	SoAdIf {SOAD_IF}		
Description	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).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00145 :		
Name	SoAdIfTriggerTransmit {SOAD_IF_TRIGGER_TRANSMIT}		
Description	Specifies if the BSW module supports the TriggerTransmit API or not. Value true means that the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local
---------------------------	--------------

SWS Item	ECUC_SoAd_00106 :		
Name	SoAdIfTxConfirmation {SOAD_IF_TX_CONFIRMATION}		
Description	Specifies if the BSW module supports the TxConfirmation API or not. Value true means that the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00143 :		
Name	SoAdLocalIpAddrAssignmentChg {SOAD_LOCAL_IP_ADDR_ASSIGNMENT_CHG}		
Description	Specifies if the BSW module supports the LocalIpAddrAssignmentChg API or not. Value true means that the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00107 :		
Name	SoAdSoConModeChg {SOAD_SO_CON_MODE_CHG}		
Description	Specifies if the BSW module supports the SoConModeChg API or not. Value true means that the API is supported.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00105 :		
Name	SoAdTp {SOAD_TP}		
Description	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).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local
---------------------------	--------------

SWS Item	ECUC_SoAd_00128 :		
Name	SoAdUseCallerInfix {SOAD_USE_CALLER_INFIX}		
Description	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().		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00129 :		
Name	SoAdUseTypeInfix {SOAD_USE_TYPE_INFIX}		
Description	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().		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00124 :		
Name	SoAdBswModuleRef		
Description	<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 <code>_SoAd<If Tp><function></code> 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>		
Multiplicity	1		
Type	Foreign reference to [ECUC-MODULE-CONFIGURATION-VALUES]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local
---------------------------	--------------

No Included Containers

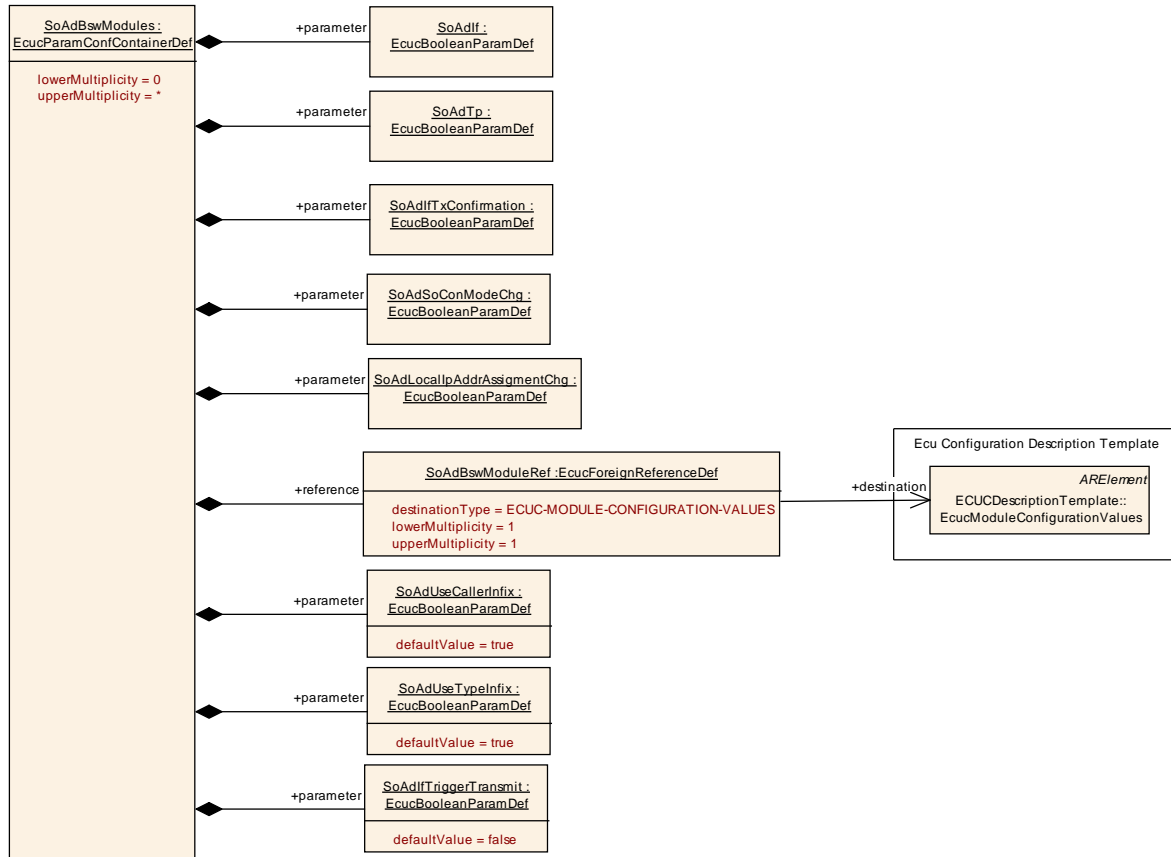


Figure 4: SoAd BswModules container

10.2.4 SoAdGeneral

SWS Item	ECUC_SoAd_0003 :
Container Name	SoAdGeneral
Description	This container contains all global configuration parameters of SoAd.
Configuration Parameters	

SWS Item	ECUC_SoAd_0002 :		
Name	SoAdDevErrorDetect {SOAD_DEV_ERROR_DETECT}		
Description	Pre-processor switch for enabling development error detection support.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local
---------------------------	--------------

SWS Item	ECUC_SoAd_00039 :		
Name	SoAdIPv6AddressEnabled {SOAD_IPv6_ADDRESS_ENABLED}		
Description	Allows for increased memory allocation to store IPv6 addresses. TRUE: Enables support for IPv6 addresses FALSE: Only IPv4 addresses are supported		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_SoAd_00062 :		
Name	SoAdMainFunctionPeriod {SOAD_MAINFUNCTION_PERIOD}		
Description	Determines the frequency at which the SoAd_MainFunction() is called in [s].		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	ECUC_SoAd_00127 :		
Name	SoAdRoutingGroupMax {SOAD_ROUTINGGROUP_MAX}		
Description	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.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00126 :		
Name	SoAdSoConMax {SOAD_SOCON_MAX}		
Description	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.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local		
SWS Item	ECUC_SoAd_00004 :		
Name	SoAdVersionInfoApi {SOAD_VERSION_INFO_API}		
Description	Activates the SoAd_GetVersionInfo() API. TRUE: Enables the SoAd_GetVersionInfo() API. FALSE: SoAd_GetVersionInfo() API is not included.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.5 SoAdConfig

SWS Item	ECUC_SoAd_00103 :		
Container Name	SoAdConfig [Multi Config Container]		
Description	This container contains the configuration parameters and sub containers of the AUTOSAR SoAd module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.		
Configuration Parameters			

SWS Item	ECUC_SoAd_00125 :		
Name	SoAdConfigurationId		
Description	Identification of the SoAd configuration.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: ECU
---------------------------	------------

Included Containers		
Container Name	Multiplicity	Scope / Dependency
SoAdPduRoute	1..*	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission.
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	1..*	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.

10.2.6 SoAdSocketConnectionGroup

SWS Item	ECUC_SoAd_00130 :
Container Name	SoAdSocketConnectionGroup
Description	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.
Configuration Parameters	

SWS Item	ECUC_SoAd_00131 :		
Name	SoAdPduHeaderEnable {SOAD_PDU_HEADER_ENABLE}		
Description	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		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local
---------------------------	--------------

SWS Item	ECUC_SoAd_00067 :		
Name	SoAdResourceManagementEnable {SOAD_RESOURCE_MANAGEMENT_ENABLE}		
Description	Enables the resource management option for this socket. May not be activated for UDP sockets in receive. TRUE: resource management option enabled FALSE: resource management option disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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

SWS Item	ECUC_SoAd_00138 :		
Name	SoAdSocketFramePriority {SOAD_SOCKET_FRAME_PRIORITY}		
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 field. 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	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local
---------------------------	--------------

SWS Item	ECUC_SoAd_00112 :		
Name	SoAdSocketIpAddrAssignmentChgNotification {SOAD_SOCKET_IP_ADDR_ASSIGN_CHG_NOTIFICATION}		
Description	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.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00018 :		
Name	SoAdSocketLocalPort {SOAD_SOCKET_LOCAL_PORT}		
Description	Local UDP or TCP port used for this connection.If this parameter set to 0 SoAd requests Tcplp to select an ephemeral port.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	0		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00137 :		
Name	SoAdSocketMsgAcceptanceFilterEnabled {SOAD_SOCKET_MSG_ACCEPTANCE_FILTER_ENABLED}		
Description	Specifies if the message acceptance filter is enabled (TRUE) or not (FALSE). Note: if a wildcard is used in SoAdSocketRemoteAddress AND SoAdSocketUdpListenOnly is FALSE, this parameter must be TRUE. Note: if multiple SoAdSocketConnections are configured for one SoAdSocketConnectionGroup, this parameter must be TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	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: SoAdSocketRemoteAddress, SoAdSocketUdpListenOnly, SoAdSocketConnectionGroup
---------------------------	---

SWS Item	ECUC_SoAd_00111 :		
Name	SoAdSocketSoConModeChgNotification {SOAD_SOCKET_SO_CON_MODE_CHG_NOTIFICATION}		
Description	Specifies if the SoCon mode change notification callback function of the upper layer shall be called in case of SoCon mode change.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00134 :		
Name	SoAdSocketTpRxBufferMin {SOAD_SOCKET_TP_RX_BUFFER_MIN}		
Description	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.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00017 :		
Name	SoAdSocketLocalAddressRef {SOAD_SOCKET_LOCAL_ADDRESS}		
Description	Local IP address and interface used for this connection.		
Multiplicity	1		
Type	Symbolic name reference to [TcpIpLocalAddr]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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

10.2.7 SoAdSocketConnection

SWS Item	ECUC_SoAd_00009 :
Container Name	SoAdSocketConnection
Description	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.
Configuration Parameters	

SWS Item	ECUC_SoAd_00016 :		
Name	SoAdSocketId {SOAD_SOCKET_ID}		
Description	Socket connection identifier used as SoConId in the interaction with upper layers.		
Multiplicity	1		
Type	EcuIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
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().

10.2.8 SoAdSocketProtocol

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

Container Choices		
Container Name	Multiplicity	Scope / Dependency
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.

10.2.9 SoAdSocketUdp

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

SWS Item	ECUC_SoAd_00024 :		
Name	SoAdSocketUdpListenOnly {SOAD_SOCKET_UDP_LISTEN_ONLY}		
Description	Specifies if the socket connection group is only used for reception (TRUE) or used for both reception and transmission (FALSE).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	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: SoAdSocketProtocol		

SWS Item	ECUC_SoAd_00132 :		
Name	SoAdSocketUdpRetryEnabled {SOAD_SOCKET_UDP_RETRY_ENABLED}		
Description	Specifies if an UdpTxBuffer shall be used (TRUE) to retry an UDP transmission in case of TCPIP_E_ARP_CACHE_MISS or not (FALSE).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	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: SoAdSocketProtocol		

SWS Item	ECUC_SoAd_00133 :		
Name	SoAdSocketUdpTriggerTimeout {SOAD_SOCKET_UDP_TRIGGER_TIMEOUT}		
Description	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.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	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: SoAdTxUdpTriggerMode		
SWS Item	ECUC SoAd 00135 :		
Name	SoAdSocketnPduUdpTxBufferMin {SOAD_SOCKET_NPDU_UDP_TX_BUFFER_MIN}		
Description	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 or an upper layer with IF API with UDP transmit retry (for single PDUs) configured, the required buffer size can be determined automatically. This optional parameter is only relevant if a nPduUdpTxBuffer is used.		
Multiplicity	0..1		
Type	EcuclIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	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: SoAdTxUdpTriggerMode		

No Included Containers

10.2.10 SoAdSocketTcp

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

SWS Item	ECUC_SoAd_00147 :		
Name	SoAdSocketTcplmmediateTpTxConfirmation {SOAD_SOCKET_TCP_IMMEDIATE_TP_TX_CONFIRMATION}		
Description	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 Tcplp.		
Multiplicity	1		
Type	EcuclBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: ECU
---------------------------	------------

SWS Item	ECUC_SoAd_00022 :		
Name	SoAdSocketTcpInitiate {SOAD_SOCKET_TCP_INITIATE}		
Description	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.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00148 :		
Name	SoAdSocketTcpKeepAlive {SOAD_SOCKET_TCP_KEEPALIVE}		
Description	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.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	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: TcpIpTcpKeepAliveEnabled		

SWS Item	ECUC_SoAd_00023 :		
Name	SoAdSocketTcpNoDelay {SOAD_SOCKET_TCP_NODELAY}		
Description	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.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	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: TcplpTcpNagleEnabled
---------------------------	--

SWS Item	ECUC SoAd 00142 :		
Name	SoAdSocketTcpTxQuota {SOAD_SOCKET_TCP_TX_QUOTA}		
Description	<p>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 Tcplp module for each socket connection of this socket connection group.</p> <p>Rationale: prohibits that a socket connection consumes all available transmit buffers at the Tcplp and blocks transmissions via other socket connections. If the optional parameter is not enabled, the amount of data is not limited.</p>		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	--		
ConfigurationClass	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

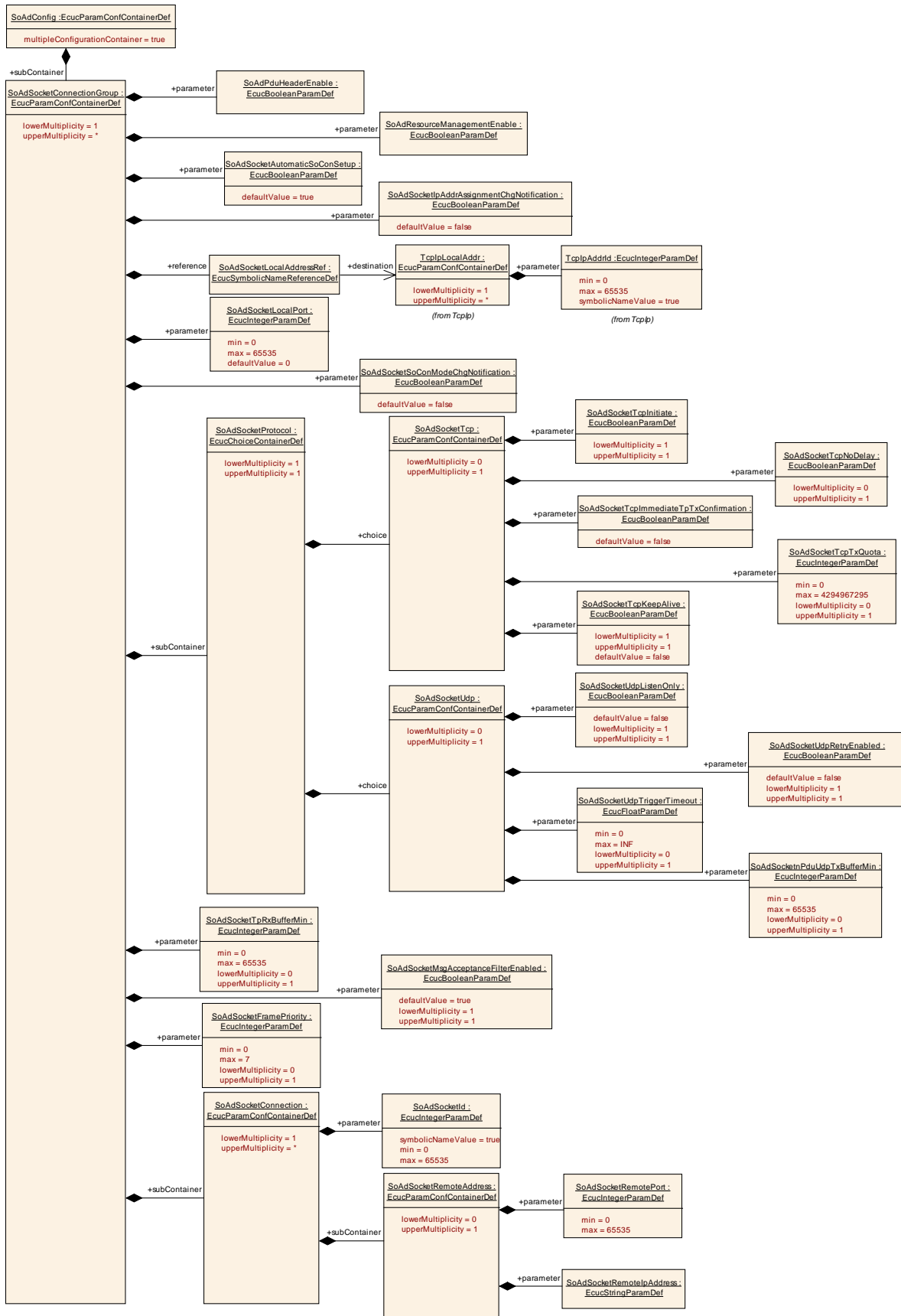


Figure 5: Socket Connection Group configuration

10.2.11 SoAdSocketRemoteAddress

SWS Item	ECUC_SoAd_00113 :
Container Name	SoAdSocketRemoteAddress
Description	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().
Configuration Parameters	

SWS Item	ECUC_SoAd_00019 :		
Name	SoAdSocketRemotelpAddress {SOAD_SOCKET_REMOTE_IP_ADDRESS}		
Description	IP address of remote node. To accept any remote IP address, set SoAdSocketRemotelpAddress to "ANY". See message acceptance policy for more details.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00020 :		
Name	SoAdSocketRemotePort {SOAD_SOCKET_REMOTE_PORT}		
Description	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.		
Multiplicity	1		
Type	EcuIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	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

10.2.12 SoAdSocketRoute

SWS Item	ECUC_SoAd_00008 :
Container Name	SoAdSocketRoute
Description	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.
Configuration Parameters	

SWS Item	ECUC_SoAd_00036 :		
Name	SoAdRxPduHeaderId {SOAD_RX_PDUHEADER_ID}		
Description	ID contained in the packet received on the TCP/IP connection if the PDU header option is enabled.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967296		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00035 :		
Name	SoAdRxSocketConnectionRef {SOAD_SOCON}		
Description	Connection on which the PDU was received. This references an entry in the Socket Connection Table.		
Multiplicity	1		
Type	Reference to [SoAdSocketConnection]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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

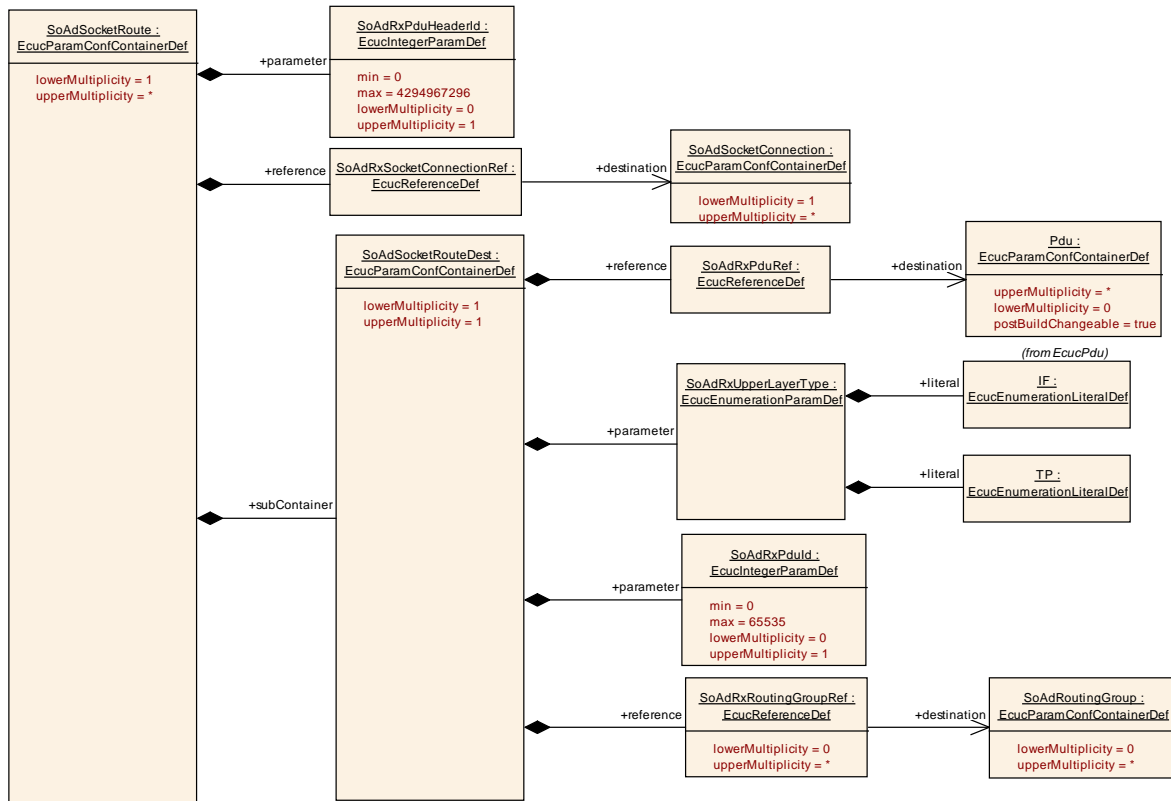


Figure 6: Socket Route configuration

10.2.13 SoAdSocketRouteDest

SWS Item	ECUC_SoAd_00114 :		
Container Name	SoAdSocketRouteDest		
Description	Describes the upper layer destination PDU for a message received on a Tcplp socket.		
Configuration Parameters			

SWS Item	ECUC_SoAd_00116 :		
Name	SoAdRxPduId {SOAD_RX_PDU_ID}		
Description	This unique identifier is used for a receive cancellation request from an upper layer of the SoAd.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: ECU
---------------------------	------------

SWS Item	ECUC_SoAd_00115 :		
Name	SoAdRxUpperLayerType {SOAD_UPPER_LAYER_TYPE}		
Description	Specifies the upper layer interface type (must be "IF" in case of multiple RxPdu).		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	IF		If interface
	TP		TP interface
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_SoAd_00038 :		
Name	SoAdRxPduRef {SOAD_RX_PDU_REF}		
Description	Reference to the global PDU structure		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_SoAd_00117 :		
Name	SoAdRxRoutingGroupRef {SOAD_RX_ROUTING_GROUP_REF}		
Description	Reference to the routing group.		
Multiplicity	0..*		
Type	Reference to [SoAdRoutingGroup]		
ConfigurationClass	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

10.2.14 SoAdPduRoute

SWS Item	ECUC_SoAd_00007 :		
Container Name	SoAdPduRoute		
Description	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission.		
Configuration Parameters			

SWS Item	ECUC_SoAd_00031 :		
Name	SoAdTxPduId {SOAD_TX_PDU_ID}		
Description	Tx PDU ID of the PDU coming from the PDU Router.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_SoAd_00118 :		
Name	SoAdTxUpperLayerType {SOAD_UPPER_LAYER_TYPE}		
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
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	ECUC_SoAd_00030 :		
Name	SoAdTxPduRef {SOAD_TX_PDU_REF}		
Description	Reference to the global PDU structure		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

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

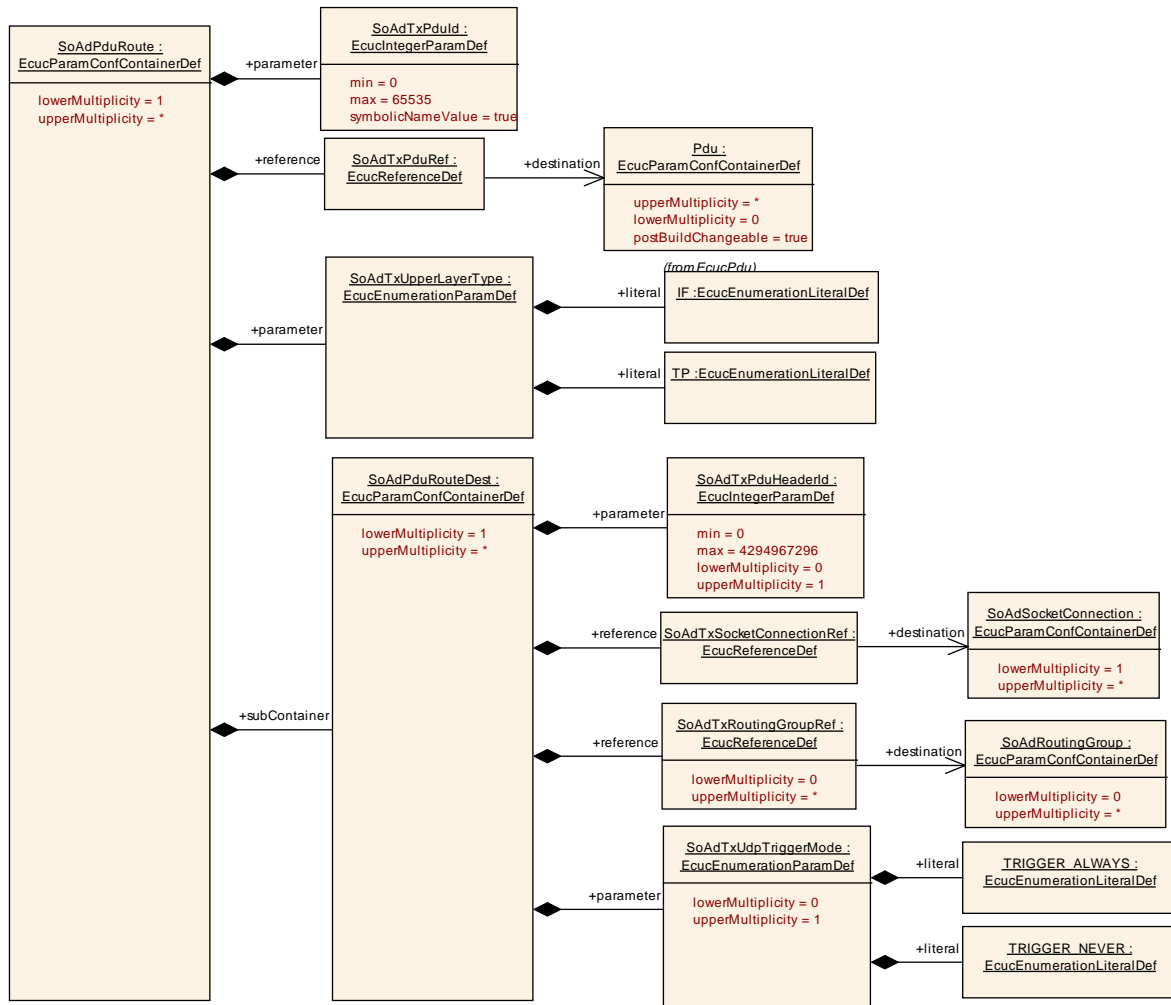


Figure 7: Pdu Route configuration

10.2.15 SoAdPduRouteDest

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

SWS Item	ECUC_SoAd_00120 :	
Name	SoAdTxPduHeaderId {SOAD_TX_PDUHEADER_ID}	
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	--	
ConfigurationClass	Pre-compile time	X VARIANT-PRE-COMPILE
	Link time	X VARIANT-LINK-TIME

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00136 :		
Name	SoAdTxUdpTriggerMode {SOAD_TX_UDP_TRIGGER_MODE}		
Description	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.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	TRIGGER_ALWAYS	PDU triggers the transmission	
	TRIGGER_NEVER	PDU does not trigger the transmission	
ConfigurationClass	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: SoAdSocketProtocol, SoAdTxUpperLayerType		

SWS Item	ECUC_SoAd_00123 :		
Name	SoAdTxRoutingGroupRef {SOAD_TX_ROUTING_GROUP_REF}		
Description	Reference to the routing group.		
Multiplicity	0..*		
Type	Reference to [SoAdRoutingGroup]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_SoAd_00034 :		
Name	SoAdTxSocketConnectionRef {SOAD_TX_SOCKET_CONNECTION_REF}		
Description	Connection on which the PDU is to be sent on, references the appropriate entry in the Socket Connection Table.		
Multiplicity	1		
Type	Reference to [SoAdSocketConnection]		
ConfigurationClass	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

10.2.16 SoAdRoutingGroup

SWS Item	ECUC_SoAd_00109 :		
Container Name	SoAdRoutingGroup		
Description	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.		
Configuration Parameters			

SWS Item	ECUC_SoAd_00121 :		
Name	SoAdRoutingGroupId {SOAD_ROUTINGGROUP_ID}		
Description	Unique ID of Routing Group		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

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

SWS Item	ECUC_SoAd_00146 :		
Name	SoAdRoutingGroupTxTriggerable {SOAD_ROUTINGGROUP_TX_TRIGGERABLE}		
Description	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).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	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, BSW00434,
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_00326, 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_00355,
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)