Document Title	Specification of Socket Adaptor
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	416
Document Classification	Standard
Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History

Release	Changed by	Change Description
4.2.2	AUTOSAR Release Management	• Clarifications and corrections of requirements • Editorial changes
4.2.1	AUTOSAR Release Management	• Introduction of IPv6 for in-vehicle communication • Support for Service Migration of Service Discovery Clients (SpecificRoutingGroup Handling) • SoAd_RequestIpAddrAssignment API extension • Clarifications and corrections of requirements and sequence charts
4.1.3	AUTOSAR Release Management	• TP API: Harmonization of ChangeParameter function • Clarifications and corrections of requirements and sequence charts • Editorial changes
4.1.2	AUTOSAR Release Management	• TP API: NotifResultType replaced by Std_ReturnType • Clarifications and corrections of requirements and sequence charts • Editorial changes • Removed chapter(s) on change documentation
4.1.1	AUTOSAR Administration	• APIs for SWS TCP/IP module interaction added/updated • Functionality to trigger IPdus for sending has been added • DoIP functionality removed
4.0.3	AUTOSAR Administration	• Rectify inconsistencies in synchronicity and reentrancy • Adjust parameter multiplicity • New traceability mechanism

Document Change History		
Release	**Changed by**	**Change Description**
3.1.5	AUTOSAR Administration	• ComStack Harmonization. • Allow for Post-Build Configuration • API for IP address change notification • Allow full handling of TCP connections
3.1.4	AUTOSAR Administration	• Initial Release Revision

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview 7
2 Acronyms and abbreviations 9
3 Related documentation 10
3.1 Input documents 10
3.2 Related standards and norms 11
3.3 Related specification 11
4 Constraints and assumptions 12
4.1 Limitations 12
4.2 Applicability to car domains 12
5 Dependencies on other modules 13
5.1 AUTOSAR TCP/IP Stack 13
5.2 Generic Upper Layer 13
5.3 File structure 13
5.3.1 Code file structure 13
5.3.2 Header file structure 13
5.4 Version check 15
6 Requirements traceability 16
7 Functional specification 25
7.1 Socket Connections 25
7.1.1 Socket Connection Open 26
7.1.2 Socket Connection Close 29
7.1.3 Socket Connection Open/Close Sequence Remarks 31
7.1.4 Notifications 32
7.2 PDU Transmission 32
7.2.1 PDU Transmission via IF-API 33
7.2.2 PDU Transmission via IF-API and nPduUdpTxBuffer 34
7.2.3 PDU Transmission via IfRoutingGroupTransmit API 35
7.2.4 PDU Transmission via TP-API 36
7.3 PDU Header option 38
7.4 PDU Reception 38
7.4.1 PDU Reception via IF-API 40
7.4.2 PDU Reception via TP-API (PDU Header disabled) 40
7.4.3 PDU Reception via TP-API (PDU Header enabled) 41
7.5 Best Match Algorithm 42
7.6 Message Acceptance Policy 43
7.7 TP PDU Cancelation 44
7.8 Disconnection and recovery 44
7.9 Routing Groups 45
7.10 PDU fan-out 46
7.11 Resource Management Option 46
7.12 Buffer handling 48
7.13 Error classification 49
7.13.1 Development Errors 49
7.13.2 Runtime Errors 49

7.13.3 Transient Faults 49
7.13.4 Production Errors 49
7.13.5 Extended Production Errors 49
7.14 Application notes 50
7.15 Debugging Concept 50
7.16 Version checking 50
8 API specification 51
8.1 Imported types 51
8.2 Type definitions 51
8.2.1 SoAd_SoConIdType 51
8.2.2 SoAd_SoConModeType 52
8.2.3 SoAd_RoutingGroupIdType 52
8.2.4 SoAd_ConfigType 52
8.3 Function definitions 53
8.3.1 General 53
8.3.2 Normal Operation 54
8.3.3 Transmit/Receive Cancelation API 57
8.3.4 Information and Control API 59
8.4 Call-back notifications 75
8.4.1 SoAd_RxIndication 75
8.4.2 SoAd_CopyTxData 76
8.4.3 SoAd_TxConfirmation 77
8.4.4 SoAd_TcpAccepted 78
8.4.5 SoAd_TcpConnected 79
8.4.6 SoAd_TcpIpEvent 79
8.4.7 SoAd_LocalIpAddrAssignmentChg 80
8.5 Scheduled functions 81
8.5.1 Terms and definitions 81
8.5.2 SoAd_MainFunction 81
8.6 Expected Interfaces 82
8.6.1 Mandatory Interfaces 82
8.6.2 Optional Interfaces 84
8.6.3 Configurable interfaces 84
9 Sequence diagrams and Transition Tables 91
9.1 Address – Assignment 91
9.2 Socket Connection Setup – UDP 92
9.3 Socket Connection Setup – TCP 94
9.4 Reception – Upper Layer If API 96
9.5 Reception – Upper Layer TP API 97
9.6 Transmission – Upper Layer If API – TCP 99
9.7 Transmission – Upper Layer If API – UDP 101
9.8 Transmission – Upper Layer TP API 102
10 Configuration specification 104
10.1 How to read this chapter 104
10.2 Containers and configuration parameters 105
10.2.1 Variants 105
10.2.2 SoAd 106
10.2.3 SoAdBswModules 106

10.2.4 SoAdGeneral 110
10.2.5 SoAdConfig 112
10.2.6 SoAdSocketConnectionGroup 112
10.2.7 SoAdSocketConnection 116
10.2.8 SoAdSocketProtocol 117
10.2.9 SoAdSocketUdp 117
10.2.10 SoAdSocketTcp 119
10.2.11 SoAdSocketRemoteAddress 124
10.2.12 SoAdSocketRoute 125
10.2.13 SoAdSocketRouteDest 127
10.2.14 SoAdPduRoute 128
10.2.15 SoAdPduRouteDest 131
10.2.16 SoAdRoutingGroup 133
10.3 Published Information 135
11 Not applicable requirements 136

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 TcpIp SWS as shown in Figure 1. The internal functional structure of the TCP/IP stack is shown schematically for information purposes.

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

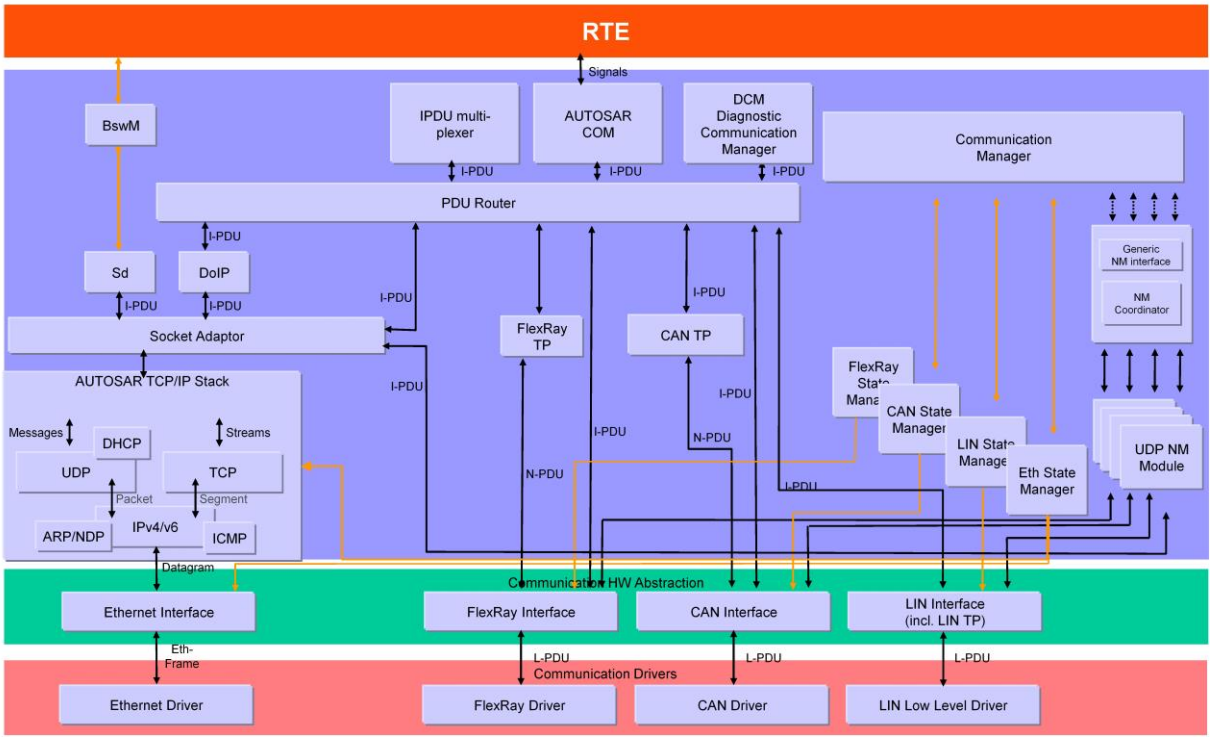


Figure 1: Extended AUTOSAR Communication Stack.

2 Acronyms and abbreviations

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

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

3 Related documentation

3.1 Input documents

[1] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[2] General Requirements on Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[3] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[4] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[5] Specification of BSW Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

[6] Specification of Default Error Tracer
AUTOSAR_SWS_DefaultErrorTracer.pdf

[7] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[8] Specification of UDP Network Management
AUTOSAR_SWS_UDPNetworkManagement.pdf

[9] Requirements on Ethernet
AUTOSAR_SRS_Ethernet.pdf

[10] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList

[11] Specification of Service Discovery
AUTOSAR_SWS_ServiceDiscovery.pdf

[12] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[13] Specification of TCP/IP Stack
AUTOSAR_SWS_TCPIP.pdf

[14] Specification of Module XCP
AUTOSAR_SWS_XCP.pdf

[15] Specification of Diagnostics over IP
AUTOSAR_SWS_DoIP.pdf

[16] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

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

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

3.3 Related specification

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

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

4 Constraints and assumptions

4.1 Limitations

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

To avoid further protocol overhead, the use of a single socket connection per PDU is described here. However, this solution is very resource intensive, particularly if many small PDUs are to be transmitted. One solution described here as an option is to add a small PDU header, containing an ID and length information. This enables transmission of multiple PDUs via one socket connection. Additionaly 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 TcpIp module implements the main protocols of the TCP/IP protocol family (TCP, UDP, IPv4, ARP, ICMP, DHCP, IPv6, NDP, ICMPv6, DHCPv6) and provides dynamic, socket based communication via Ethernet. The SoAd module is one of the possible upper layer modules of the TcpIp module.

5.2 Generic Upper Layer

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

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

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

5.3 File structure

5.3.1 Code file structure

For details refer to the chapter 5.1.6 “Code file structure” in *SWS_BSWGeneral.*

5.3.2 Header file structure

[SWS_SoAd_00072] ⌈The SoAd module shall provide the following H-files:

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

[SWS_SoAd_00073] ⌈The SoAd module shall include the following H-files of other modules:

- `TcpIp.h` – header file of the AUTOSAR TCP/IP stack
- `ComStack_Types.h` [3]
- `<UL>_Cbk.h` (for callback functions of the SoAd upper layer modules) ⌋( )

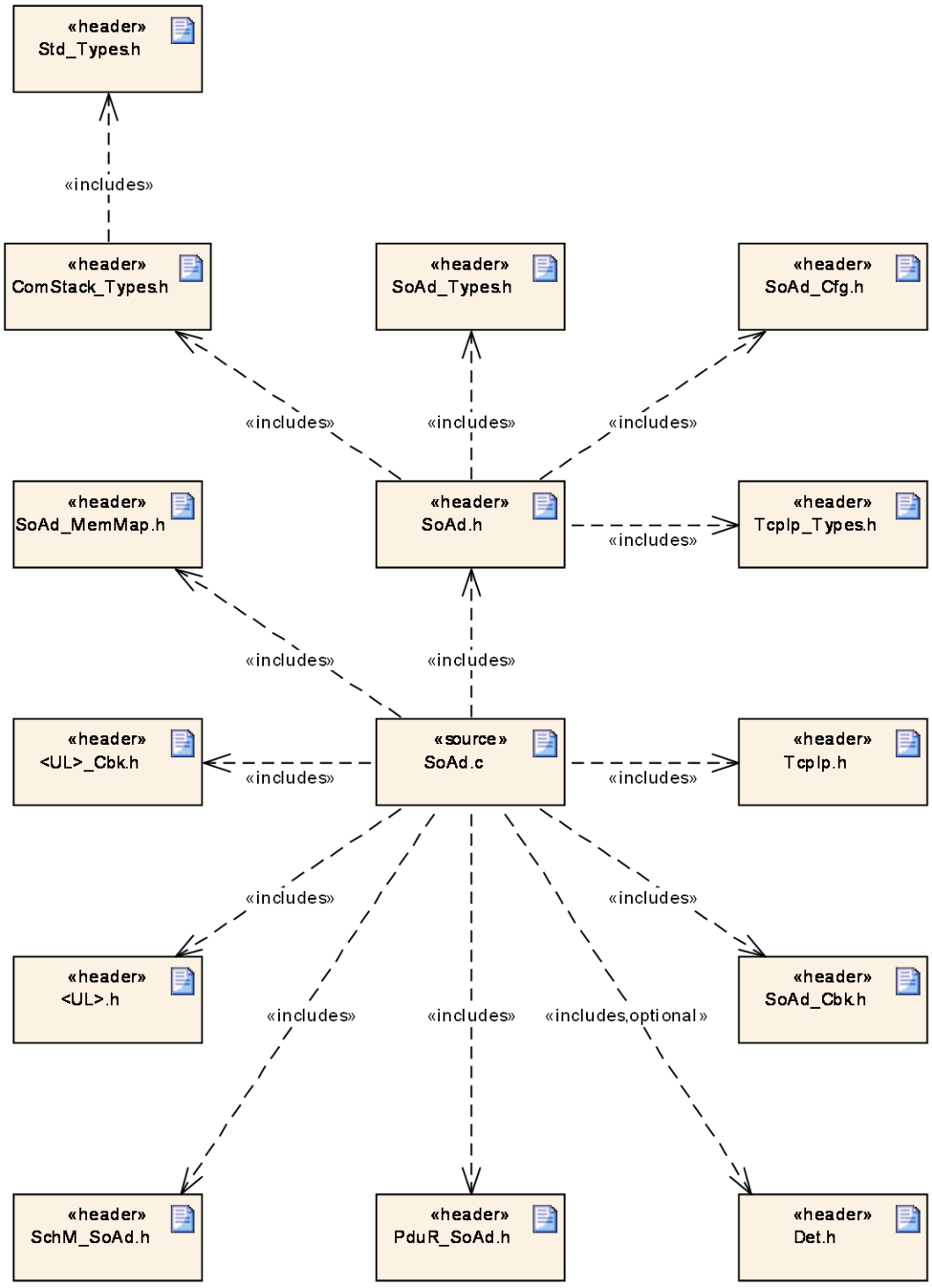


Figure 2: AUTOSAR SoAd header file structure.

5.4 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_SoAd_00001
-	-	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_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_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_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
-	-	SWS_SoAd_00693
-	-	SWS_SoAd_00696
-	-	SWS_SoAd_00697
-	-	SWS_SoAd_00698
-	-	SWS_SoAd_00699
-	-	SWS_SoAd_00700
-	-	SWS_SoAd_00701
-	-	SWS_SoAd_00702
-	-	SWS_SoAd_00703
-	-	SWS_SoAd_00704
-	-	SWS_SoAd_00705

-	-	SWS_SoAd_00706
-	-	SWS_SoAd_00707
-	-	SWS_SoAd_00708
-	-	SWS_SoAd_00709
-	-	SWS_SoAd_00710
-	-	SWS_SoAd_00711
-	-	SWS_SoAd_00712
-	-	SWS_SoAd_00713
-	-	SWS_SoAd_00714
-	-	SWS_SoAd_00715
-	-	SWS_SoAd_00716
-	-	SWS_SoAd_00717
-	-	SWS_SoAd_00718
-	-	SWS_SoAd_00719
-	-	SWS_SoAd_00720
-	-	SWS_SoAd_00721
-	-	SWS_SoAd_00722
-	-	SWS_SoAd_00723
-	-	SWS_SoAd_00724
-	-	SWS_SoAd_00728
SRS_BSW_00005	Modules of the ÂµC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_SoAd_00296
SRS_BSW_00006	The source code of software modules above the ÂµC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_SoAd_00296
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_SoAd_00296
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_SoAd_00296
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_SoAd_00296
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_SoAd_00296
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_SoAd_00296
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_SoAd_00296
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_SoAd_00296
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the	SWS_SoAd_00296

	strategy used in the system	
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_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_00344	BSW Modules shall support link-time configuration	SWS_SoAd_00726
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_SoAd_00725
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_SoAd_00296
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_SoAd_00296
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_SoAd_00727
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_SoAd_00727
SRS_BSW_00410	Compiler switches shall have defined values	SWS_SoAd_00296
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_SoAd_00296
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_SoAd_00296
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_SoAd_00296
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_SoAd_00296
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_SoAd_00296

SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_SoAd_00296
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_SoAd_00296
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_SoAd_00296
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_SoAd_00296
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_SoAd_00296
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_SoAd_00296
SRS_Eth_00085	Robustness aganist the change of logical addresses	SWS_SoAd_00694, SWS_SoAd_00695

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 opend automatically or manually via a request from the upper layer. For disconnection and recovery of a socket connection a policy is defined. An upper layer of the SoAd can use the IF-API as well as the TP-API for transmission and reception of PDUs.To selectively enable/disable the routing of PDUs from or to socket connections PDU routing groups are defined and can be controlled by the upper layer of the SoAd. An IF-PDU can also be forwarded to multiple socket connections or a message received from a socket connection can be forwarded as different IF-PDUs to the same or different upper layers of the SoAd (PDU Fan-out).

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

7.1 Socket Connections

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

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

[SWS_SoAd_00588][SoAd shall store a request to open or close a socket connection when called with `SoAd_OpenSoCon()` and `SoAd_CloseSoCon()` respectively, but handle the request only in the `SoAd_MainFunction()` respecting the connection setup and shutdown policy.] ()

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_LocalIpAddrAssignmentChg()` 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_SoAdGetSocket()` with the TcpIp_DomainType implicitly specified by SoAdSocketLocalAddressRef, and the protocol type specified by SoAdSocketProtocol.
(2) Change the socket specific parameters according to [SWS_SoAd_00689]
(3) Bind the socket to the local address and port by calling `TcpIp_Bind()` with the local address identifier specified by SoAdSocketLocalAddressRef and local port specified by SoAdSocketLocalPort.
(4) In case of a TCP socket initiate the TCP connection by calling TcpIp_TcpConnect().

] ()

[SWS_SoAd_00638][SoAd shall perform the following actions within `SoAd_MainFunction()` to open a TCP socket connection with SoAdSocketTcpIni-tiate set to FALSE:

(1) In case no Listen-Socket is assigned to the socket connection:
    (a) Get an appropriate socket from TcpIp by calling `TcpIp_SoAdGetSocket()` with the TcpIp_DomainType implicitly specified by SoAdSocketLocalAddressRef, and the protocol type specified by SoAdSocketProtocol.
    (b) Change the socket specific parameters according to [SWS_SoAd_00689]

(c) Bind the socket to the local address and port by calling `TcpIp_Bind()` with the local address identifier specified by SoAdSocketLocalAddressRef and local port specified by SoAdSocketLocalPort.
(d) Assign the Listen-Socket to the socket connection group
(e) Activate the socket connection to accept connections from remote nodes
(f) Listen for a remote connection requests on the Listen-Socket by calling TcpIp_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 SoAdSocket-TcpInitiate set to FALSE) share one TcpIp socket for incoming connection requests ("Listen-Socket"), but use a separate TcpIp socket created by the TcpIp 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 TcpIp by calling `TcpIp_SoAdGetSocket()` with the domain type implicitly specified by SoAdSocketLocalAddressRef, and the protocol type specified by SoAdSocketProtocol.
(b) Change the socket specific parameters according to [SWS_SoAd_00689]
(c) Bind the socket to the local address and port by calling `TcpIp_Bind()` with the local address identifier specified by SoAdSocketLocalAddressRef and local port specified by SoAdSocketLocalPort.
(d) Assign the UDP socket to the socket connection group
(e) Activate the socket connection for communication via the shared UDP socket of the socket connection group

(2) In case the UDP socket is already assigned to the socket connection group:
(a) Activate the socket connection for communication via the shared UDP socket of the socket connection group

] ()

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

[SWS_SoAd_00689][In case socket related parameters shall be changed as part of allocating a new socket, SoAd shall change the parameters according to the configuration of the associated socket connection by calling `TcpIp_ChangeParameter()` with ParameterId and ParameterValue for each related clause as specified below:

(1) In case of a TCP socket: TCPIP_PARAMID_TCP_RXWND_MAX and the value specified by SoAdSocketTpRxBufferMin if the optional parameter is enabled

(2) TCPIP_PARAMID_FRAMEPRIO and the value specified by SoAdSocketFramePriority if the optional parameter is enabled
(3) In case of a TCP socket: TCPIP_PARAMID_TCP_NAGLE and the value 0x01 if the related optional parameter SoAdSocketTcpNoDelay is set to FALSE or 0x00 if the parameter is set to TRUE.
(4) In case of a TCP socket: TCPIP_PARAMID_TCP_KEEPALIVE and the value specified by SoAdSocketTcpKeepAlive
(5) In case of a TCP socket: TCPIP_PARAMID_TCP_KEEPALIVE_TIME and the value specified by SoAdSocketTcpKeepAliveTime if the optional parameter is enabled
(6) In case of a TCP socket: TCPIP_PARAMID_TCP_KEEPALIVE_PROBES_MAX and the value specified by SoAdSocketTcpKeepAliveProbesMax if the optional parameter is enabled
(7) In case of a TCP socket: TCPIP_PARAMID_TCP_KEEPALIVE_INTERVAL and the value specified by SoAdSocketTcpKeepAliveInterval if the optional parameter is enabled.⌋ ()

[SWS_SoAd_00591]⌈ Within SoAd_MainFunction() and after successfully performing the open actions, SoAd shall change the state of the socket connection to SOAD_SOCON_ONLINE in case of a UDP socket and either SoAdSocketUdpListenOnly is set to TRUE or a remote address is set by a value that does not contain any wildcards.⌋ ()

[SWS_SoAd_00686]⌈ Within SoAd_MainFunction() and after successfully performing the open actions, SoAd shall change the state of the socket connection to SOAD_SOCON_RECONNECT in case of
(1) a TCP socket connection or
(2) a UDP socket connection that is configured with a remote address containing wildcards.⌋ ()

[SWS_SoAd_00592]⌈ Within SoAd_RxIndication() and before analyzing or forwarding of any message data, SoAd shall (a) overwrite the remote address parts specified with wildcards (e.g. remote IP address set to TCPIP_IPADDR_ANY) with the related source address parts of the received message and (b) change the state of the socket connection to SOAD_SOCON_ONLINE in case all of the following conditions are true:
(1) Current connection state is not SOAD_SOCON_ONLINE
(2) UDP socket
(3) SoAdSocketUdpListenOnly is set to FALSE
(4) SoAdSocketMsgAcceptanceFilterEnabled is set to TRUE
(5) Remote address is set, but contains wildcards
(6) Received message is accepted according to the message acceptance policy
⌋ ()

[SWS_SoAd_00593]⌈ Within SoAd_TcpConnected() SoAd shall change the state of the socket connection to SOAD_SOCON_ONLINE in case all of the following conditions are true:
(1) Current connection state is not SOAD_SOCON_ONLINE
(2) TCP socket

(3) SoAdSocketTcpInitiate is set to TRUE

]()

[SWS_SoAd_00594][At SoAd_TcpAccepted(), SoAd shall perform the following actions if the TCP SoAdSocketConnectionGroup related to SocketId has SoAdSocketTcpInitiate set to FALSE:

(1) choose one of the socket connections using the best match algorithm (see [SWS_SoAd_00680]), and either proceed with the selected socket connection or skip further processing and return with E_NOT_OK if no match can be found
(2) overwrite the remote address parts specified with wildcards (e.g. remote IP address set to TCPIP_IPADDR_ANY) with the related source address parts of the received message if the remote address set for the socket connection contains wildcards
(3) assign the TcpIp socket used for the established connection and provided as parameter SocketIdConnected to the chosen socket connection,
(4) change the state of this socket connection to SOAD_SOCON_ONLINE and return E_OK

]()

[SWS_SoAd_00636][At SoAd_TcpAccepted(), SoAd shall perform the following actions if the TCP SoAdSocketConnectionGroup related to SocketId has both SoAdSocketTcpInitiate and SoAdSocketMsgAcceptanceFilterEnabled set to FALSE and is not online (i.e. current connection state not SOAD_SOCON_ONLINE):

(1) assign the TcpIp socket used for the established connection and provided as parameter SocketIdConnected to the socket connection and
(2) change the state of the socket connection to SOAD_SOCON_ONLINE and return E_OK.

]()

[SWS_SoAd_00595][For socket connection with PDU Header mode disabled (SoAdPduHeaderEnable = FALSE) and an upper layer with TP-API, SoAd shall call <Up>_[SoAd][Tp]StartOfReception() with TpSduLength = 0 at the end of the connection setup.]()

7.1.2 Socket Connection Close

[SWS_SoAd_00604][In the `SoAd_MainFunction()`, SoAd shall close each socket connection which fulfills all of the following criteria:

(1) Current connection state is not SOAD_SOCON_OFFLINE
(2) Close is explicitly requested by a previous `SoAd_CloseSoCon()` call
(3) No upper layer requested to keep the socket connection open at the time of the SoAd_CloseSoCon() call (i.e. `SoAd_CloseSoCon()` has been called as often as `SoAd_OpenSoCon()`) or SoAd_CloseSoCon() has been called with abort set to TRUE.

]()

[SWS_SoAd_00637][SoAd shall perform the following actions within `SoAd_MainFunction()` to close a socket connection:
(1) Terminate active TP sessions (if any) and notify the upper layer about the termination
(2) Disable further transmission or reception for this socket connection, i.e. new transmit requests shall be rejected with E_NOT_OK and received messages shall simply be discarded.
(3) Close related TcpIp sockets
(4) Change the state of the socket connection to SOAD_SOCON_OFFLINE if closing of the socket connection results from a SoAd_CloseSoCon() request or to SOAD_SOCON_RECONNECT otherwise.
] ()

[SWS_SoAd_00640][To notify the upper layer about the termination of an active TP transmission on closing a socket connection within `SoAd_MainFunction()`, SoAd shall call `<Up>_[SoAd][Tp]TxConfirmation()` with parameter result set to
(1) E_OK if disconnect is caused by SoAd_CloseSoCon() and all data was correctly transmitted, and
(2) E_NOT_OK for any other cause.
] ()

[SWS_SoAd_00641][To notify the upper layer about the termination of an active TP reception on closing a socket connection within `SoAd_MainFunction()`, SoAd shall call `<Up>_[SoAd][Tp]RxIndication()` with parameter result set to
(1) E_OK if disconnection is caused by SoAd_CloseSoCon() and all received data was correctly delivered to the upper layer, and
(2) E_NOT_OK for any other cause.
] ()

[SWS_SoAd_00642][To close related TcpIp sockets on closing a socket connection within SoAd_MainFunction(), SoAd shall perform the following actions:
(1) In case of a TCP socket connection:
  (a) Close the related socket by calling `TcpIp_CloseSocket()` with parameter abort set to the same value as provided by `SoAd_CloseSoCon()` or set to FALSE in case closing was not initiated by `SoAd_CloseSoCon()`.
  (b) If all socket connections of a TCP socket connection group have been closed by `SoAd_CloseSoCon()`: Close the related Listen-Socket by calling `TcpIp_CloseSocket()` with parameter abort set to the same value as provided by `SoAd_CloseSoCon()` or set to FALSE in case closing was not initiated by `SoAd_CloseSoCon()`.
(2) In case of a UDP socket connection:
  (a) If the socket connection is NOT part of a socket connection group (i.e. there is only one socket connection in the socket connection group configuration container): Close the related socket by calling `TcpIp_CloseSocket()` with parameter abort set to the same value as

provided by `SoAd_CloseSoCon()` or set to FALSE in case closing was not initiated by `SoAd_CloseSoCon()`.

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

⌋ ()

[SWS_SoAd_00643]⌈ Within SoAd_TcpIpEvent() with Event set to TCPIP_UDP_CLOSED, SoAd shall
(1) remove the assignment of the TcpIp socket identified by SocketId from the related UDP socket connection group and
(2) close all socket connections of the related socket connection group that are in SOAD_SOCON_ONLINE (i.e. perform the specified closing actions with the exception of closing related TcpIp sockets)⌋ ()

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

[SWS_SoAd_00646]⌈ Within SoAd_TcpIpEvent() with Event set to TCPIP_TCP_CLOSED or TCPIP_TCP_RESET, SoAd shall
(1) remove the assignment of the TcpIp socket identified by SocketId from the related socket connection and
(2) close the socket connection if it is in SOAD_SOCON_ONLINE (i.e. perform the specified closing actions with the exception of closing related TcpIp socket). ⌋ ()

[SWS_SoAd_00688]⌈ Within SoAd_TcpIpEvent() with Event set to TCPIP_TCP_FIN_RECEIVED SoAd shall close the related socket by calling TcpIp_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>_LocalIpAddrAssignmentChg()` and the new address state if SoAdSocketIpAddrAssignmentChgNotification is set to TRUE for the socket connection.] ()

7.2 PDU Transmission

For the transmission of an upper layer module PDU via an UDP or TCP socket, the SoAd configuration specifies a PDU route which is linked to a socket connection. A PDU route (SoAdPduRoute, SoAdPduRouteDest) describes the route from an upper

layer module of the SoAd to the related socket of the TcpIp stack which is described by the socket connection (SoAdSocketConnection, SoAdSocketConnectionGroup). The upper layer module of the SoAd may use the Interface (IF) API or the Transport Protocol (TP) API for the transmit request and data provision respectively.

7.2.1 PDU Transmission via IF-API

[SWS_SoAd_00539][For the transmission of a PDU requested by an upper layer using the IF-API, the SoAd shall
(1) Identify the related socket connection and PDU route by using the 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_00542][In case of a TCP socket connection the SoAd shall call `TcpIp_TcpTransmit()` with SocketId specified in the SocketConnection, the PDU length specified in the `SoAd_IfTransmit()` call, as AvailableLength and ForceRetrieve set to TRUE.] ()

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

[SWS_SoAd_00543][The TcpIp module will retrieve the PDU data within the context of the TcpIp transmit call by using `SoAd_CopyTxData()` where the SoAd shall copy (the requested part of) the PDU to the memory specified by parameter BufPtr.] ()

[SWS_SoAd_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][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). Pdus 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_00696][SoAd shall maintain a nPduUdpTxBuffer specific timer for each socket connection with nPduUdpTxBuffer.] ()

[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 nPduUdpTxBuffer specific timer expired.] ()

[SWS_SoAd_00697][If a PDU with TriggerMode set to TRIGGER_NEVER with a specific SoAdTxUdpTriggerTimeout is transmitted SoAd shall set the nPduUdpTxBuffer specific timer to the value of SoAdTxUdpTriggerTimeout if it is lower than the current nPduUdpTxBuffer specific timer value.] ()

[SWS_SoAd_00683][If a PDU with TriggerMode set to TRIGGER_NEVER without a specific SoAdTxUdpTriggerTimeout is transmitted SoAd shall set the the nPduUdpTxBuffer specific timer to the value of SoAdSocketUdpTriggerTimeout if it is configured and lower than the current nPduUdpTxBuffer specific timer value.] ()

[SWS_SoAd_00684][SoAd shall stop the nPduUdpTxBuffer specific timer 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_00720][At SoAd_IfSpecificRoutingGroupTransmit() SoAd shall store a trigger transmit request for each SoAdPduRouteDest that contains a reference to the routing group identified by the parameter id for transmission on the socket connection identified by the parameter SoConId in the SoAd_MainFunction().] ()

[SWS_SoAd_00665][[In the SoAd_MainFunction() the SoAd shall check for pending triggered transmit request for SoAdPduRouteDest and identify all related IF-PDUs. For each identified IF-PDU SoAd shall process as specified below:

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

] ()

[SWS_SoAd_00728][To trigger PDU data from an upper layer SoAd shall set PduInfoType.SduDataPtr to the location of the buffer where the data shall be copied, set PduInfoType.SduDataLength to the length of this buffer and then call <Up>_[SoAd][If]-TriggerTransmit().] ()

7.2.4 PDU Transmission via TP-API

[SWS_SoAd_00551][For the transmission of a PDU requested by an upper layer using the TP-API, the SoAd shall
(1) Skip further processing and return with E_NOT_OK if the PDU length is 0.
(2) Identify the related socket connection and PDU route by using the SoAdSrcPduId provided at `SoAd_TpTransmit()`.
(3) Store the TP transmission request for further processing in the `SoAd_Main-Function()`.
] ()

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 TcpIp 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 TcpIp_TcpTransmit() with SocketId specified in the SocketConnection, the PDU

length set to the value returned by availableDataPtr of the previous call to <Up>_[SoAd][Tp]CopyTxData() as AvailableLength and ForceRetrieve set to FALSE.] ()

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

[SWS_SoAd_00555][In case of a UDP socket connection the SoAd shall copy (the requested part of) the PDU from the SoAd TP transmit buffer to the memory specified by parameter BufPtr within `SoAd_CopyTxData()`.
] ()

[SWS_SoAd_00556][In case of a TCP socket connection the SoAd shall forward the request to the related upper layer by calling `<Up>_[SoAd][Tp]CopyTxData()` to copy (the requested part of) the PDU to the memory specified by parameter BufPtr within `SoAd_CopyTxData()`.
] ()

[SWS_SoAd_00651][If `<Up>_[SoAd][Tp]CopyTxData()` return with BUFREQ_E_NOT_OK for a TCP socket connection, SoAd shall (a) disable further transmission or reception for this socket connection (i.e. new transmit requests shall be rejected with E_NOT_OK and received messages shall simply be discarded) and (b) close the socket connection in the next `SoAd_MainFunction()`.] ()

[SWS_SoAd_00557][In case of a UDP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation>()`) and E_OK as result within the `SoAd_MainFunction()` after `TcpIp_UdpTransmit()` returns with TCPIP_OK.] ()

[SWS_SoAd_00667][In case of a TCP socket connection configured with SoAdSocketTcpImmediateTpTxConfirmation set to TRUE the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation>()`) and E_OK as result within the `SoAd_MainFunction()` after `TcpIp_TcpTransmit()` returns E_OK.] ()

[SWS_SoAd_00670][In case of a TCP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation>()`) and E_NOT_OK as result within the `SoAd_MainFunction()` after `TcpIp_TcpTransmit()` returns with E_NOT_OK.] ()

[SWS_SoAd_00558][In case of a TCP socket connection configured with SoAdSocketTcpImmediateTpTxConfirmation set to FALSE the SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][Tp]TxConfirmation>()`) and E_OK as result within the

SoAd_TxConfirmation() callback function after all TP PDU data have been confirmed for transmission.⌋ ()

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

7.3 PDU Header option

[SWS_SoAd_00197**]**⌈ In case PDU header option is enabled (SoAdPduHeader-Enable is TRUE) for a socket connection and PDU transmission, SoAd shall insert the PDU Header with the configured HeaderId and the actual PDU length directly before the PDU data, i.e. TcpIp_UdpTransmit() or TcpIp_TcpTransmit() shall be called with a TotalLength or AvailableLength increased by the PDU Header length, the PDU Header shall be copied before the PDU data to a SoAd UDP transmit buffer (if any) and a memory specified by TcpIp within SoAd_CopyTxData() requesting the begin of the PDU data.

⌋ ( )

[SWS_SoAd_00198**]** ⌈The SoAd PDU header shall consist of a 4 byte ID field for unique identification of the PDU at the receiver and a 4 byte length field specifying the data length of the PDU. Both in BigEndian byte order.⌋ ( )

7.4 PDU Reception

For the reception of a PDU via an UDP or TCP socket, the SoAd configuration specifies a socket route which refers to a socket connection. A socket route (SoAdSocketRoute, SoAdSocketRouteDest) describes the route from an UDP or TCP socket of the TcpIp stack (which is described by the socket connection (SoAdSocketConnection, SoAdSocketConnectionGroup)) to the related upper layer module of the SoAd.
The upper layer module of the SoAd may use the Interface (IF) API or the Transport Protocol (TP) API for PDU reception.

[SWS_SoAd_00562]⌈ For the reception of a message from an UDP or TCP socket and forwarding of the received data as PDU to the related upper layer the SoAd shall

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

⌋ ()

[SWS_SoAd_00657][In case more than one socket connection belongs to a UDP socket connection group, a UDP socket is shared between all socket connections of the group and the related socket connection shall be selected according to the best match algorithm (see [SWS_SoAd_00680]).] ()

[SWS_SoAd_00563][In case PDU header option is disabled (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_00709][If `SoAdSocketUdpStrictHeaderLenCheckEnabled` is enabled SoAd shall check if the length of the received UDP message does match the accumulated length of all PDUs including their PDU headers prior to forwarding any data to an upper layer. If the their lengths are different SoAd shall silently drop the whole message without forwarding any data.] ()

[SWS_SoAd_00559][In case PDU header option is enabled (`SoAdPduHeaderEnable` is `TRUE`) for a socket connection, SoAd shall convert the message into a PDU within `SoAd_RxIndication()` according to the following:

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

] ()

[SWS_SoAd_00710][In case no valid PDU data was forwarded to an upper layer and the remote address of the socket connection was overwritten according to [SWS_SoAd_00592] in context of the same `SoAd_RxIndication()`, SoAd shall revert the remote address change and set the state of the socket connection back to `SOAD_SOCON_RECONNECT`.] ()

[SWS_SoAd_00564][In case of a TCP socket connection the SoAd shall confirm the reception of all data either forwarded to the upper layer or finally handled by SoAd (e.g. discarded data or processed PDU Header) by calling `TcpIp_TcpReceived()` within `SoAd_RxIndication()` or `SoAd_MainFunction()` respectively.] ()

[SWS_SoAd_00565][SoAd shall process TP- and IF-PDUs independently and within each type according to the received order per socket connection.] ()

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

[SWS_SoAd_00566][SoAd shall preserve the order of received data when using a SoAd receive buffer] ().

[SWS_SoAd_00693][If development error detection is enabled: Whenever a PDU or a part of a PDU is received, that has to be stored in a SoAd receive buffer, is larger than the remaining available buffer size SoAd shall shall raise the development error `SOAD_E_NOBUFS`.] ()

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]Copy-RxData()`
  (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 (newley) received data: copy all received data to the SoAd receive buffer for later processing by SoAd_MainFunction()

⌋ ()

[SWS_SoAd_00569]⌈ In the `SoAd_MainFunction()` the SoAd shall process as specified below if the SoAd receive buffer contains TP data for a socket connection with PDU Header mode disabled:

(1) Query the available amount of data at the upper layer by calling the configurable callback function `<Up>_[SoAd][Tp]CopyRxData()` with PduInfoType.SduLength = 0.

(2) Copy all data belonging to this socket connection from the SoAd receive buffer which can be accepted by the upper layer module determined at (1) to the upper layer by calling `<Up>_[SoAd][Tp]CopyRxData()`

⌋ ()

[SWS_SoAd_00570]⌈ If `<Up>_[SoAd][Tp]StartOfReception()` or `<Up>_[SoAd][Tp]CopyRxData()` return with BUFREQ_E_NOT_OK for a socket connection with PDU Header mode disabled, SoAd shall (a) disable further transmission or reception for this socket connection (i.e. new transmit requests shall be rejected with E_NOT_OK and received messages shall simply be discarded) and (b) close the socket connection in the next `SoAd_MainFunction()`.⌋ ()

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

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

[SWS_SoAd_00571]⌈ SoAd shall perform the following further actions within the `SoAd_RxIndication()` function for reception of a PDU from a socket connection with PDU Header mode enabled to an upper layer using the TP-API:

(1) if no TP reception is in progress for the related socket connection

(a) After reception of a complete PDU Header, call `<Up>_[SoAd][Tp]StartOfReception()` of the related upper layer module with RxPduId set to the ID specified by the upper layer module for the PDU referenced by SoAdRxPduRef, set TpSduLength to the length specified in the PDU Header, and set PduInfoType.SduDataPtr and PduInfoType.SduLength to provide already received PDU data to the upper layer.

(b) if not all data can be processed (i.e. forwarded to an upper layer or stored in a SoAd receive buffer), discard all received data, call `<Up>_[SoAd][Tp]RxIndication()` with the same RxPduId as used at `<Up>_[SoAd][Tp]StartOfReception()` and result set to E_NOT_OK and skip further processing

(c) call `<Up>_[SoAd][Tp]CopyRxData()` of the related upper layer module with the same RxPduId as used at `<Up>_[SoAd][Tp]StartOf-Reception()` and PduInfoType.SduDataPtr pointing to the PDU data provided by `SoAd_RxIndication()` and PduInfoType.SduLength set to

minimum of the received PDU data and the available receive buffer in the upper layer module specified by bufferSizePtr at `<Up>_[SoAd][Tp]StartOfReception()`)

(d) call `<Up>_[SoAd][Tp]RxIndication()` if the complete PDU has been forwarded to the upper layer, otherwise copy all received data which could not be forwarded to the upper layer to a SoAd receive buffer for later processing by SoAd_MainFunction()

(2) if a TP reception is in progress for the related socket connection and the related SoAd receive buffer is able to store all received data: copy all received data to the related SoAd receive buffer for later processing by `SoAd_MainFunction()`

⌋ ()

[SWS_SoAd_00572]⌈ If `<Up>_[SoAd][Tp]StartOfReception()` does not return BUFREQ_OK for a socket connection with PDU Header mode enabled, SoAd shall simply discard all data of the PDU.⌋ ()

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

[SWS_SoAd_00573]⌈ If `<Up>_[SoAd][Tp]CopyRxData()` does not return BUFREQ_OK for a socket connection with PDU Header mode enabled, SoAd shall terminate the TP receive session, simply discard all data of the PDU and call `<Up>_[SoAd][Tp]RxIndication()` with E_NOT_OK.⌋ ()

[SWS_SoAd_00574]⌈ In the `SoAd_Mainfunction()` the SoAd shall process as specified below if a TP reception is in progress for a socket connection with PDU Header mode enabled:

(1) Query the available amount of data at the upper layer by calling the configurable callback function `<Up>_[SoAd][Tp]CopyRxData()` with PduInfoType.SduLength = 0.

(2) Copy all data belonging to this socket connection from the SoAd receive buffer which can be accepted by the upper layer module determined at (1) to the upper layer by calling `<Up>_[SoAd][Tp]CopyRxData()`

(3) call `<Up>_[SoAd][Tp]RxIndication()` if the complete PDU has been forwarded to the upper layer and dispatch the next TP-PDU (if any)

⌋ ()

7.5 Best Match Algorithm

[SWS_SoAd_00680]⌈ SoAd shall use the following best match algorithm to select a socket connection of a socket connection group based on a provided (specific) remote address:

(1) socket connections that have no (specific or wildcard) remote address set shall be ignored

(2) the remote address of the remaining socket connections shall be compared with the provided remote address and the socket connection with the best match

according to the following ordered list (item listed earlier has higher priority towards items listed later) shall be selected:
(a) IP address and port match
(b) IP address match (and wildcard set for port)
(c) Port match (and wildcard set for IP address)
(d) Wildcards are used for both IP address and port
(e) No match (i.e. no socket connections can be selected)
⌋ ()

7.6 Message Acceptance Policy

[SWS_SoAd_00524]⌈ If SoAdSocketMsgAcceptanceFilterEnabled is TRUE, SoAd shall only accept TCP connections or UDP datagrams from remote nodes with a source address that matches the remote address specified in the socket connection (either via configuration parameters SoAdSocketRemoteIpAddress and SoAdSocketRemotePort or set online with `SoAd_SetRemoteAddr()` API).⌋ ()

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

[SWS_SoAd_00525]⌈ A remote address matches if both IP address and port match. The IP addresses match if they are identical or if the specified IP address is set to TCPIP_IPADDR_ANY (TCPIP_IP6ADDR_ANY). The port matches if they are identical or if the specified port is set to TCPIP_PORT_ANY.⌋ ()

[SWS_SoAd_00582]⌈ For a UDP socket connection of type automatic (i.e. configuration parameter SoAdSocketAutomaticSoConSetup set to TRUE) which uses a wildcard in the configured remote address (i.e. an ANY-String for IP address or port), SoAd shall (a) change the state of the socket connection to SOAD_SOCON_RECONNECT and (b) reset the remote address to the configured remote address after a PDU transmission, directly before the related transmit confirmation function is called (or would be called if such a function is not configured).⌋ ()

[SWS_SoAd_00644]⌈ For a TCP socket connection of type automatic (i.e. configuration parameter SoAdSocketAutomaticSoConSetup set to TRUE) which uses a wildcard in the configured remote address (i.e. an ANY-String for IP address or port), SoAd shall (a) disable further transmission or reception for this socket connection (i.e. new transmit requests shall be rejected with E_NOT_OK and received messages shall simply be discarded) after a PDU transmission, directly before the related transmit confirmation function is called (or would be called if such a function is not configured) and (b) close the socket connection in the next `SoAd_MainFunction()`.⌋ ()

[SWS_SoAd_00527]⌈ SoAd shall reset the remote address to the configured remote address (or unset the remote address in case no remote address has been configured) within `SoAd_MainFunction()` when a socket connection is closed.⌋ ()

[SWS_SoAd_00635][If SoAdSocketMsgAcceptanceFilterEnabled is FALSE, SoAd shall accept all TCP connection or UDP datagrams from remote nodes.] ()

7.7 TP PDU Cancelation

[SWS_SoAd_00575][SoAd shall store a cancellation request when called with `SoAd_TpCancelReceive()` and `SoAd_TpCancelTransmit()`, but handle the request only in the `SoAd_MainFunction()` respecting the connection loss and recovery policy.] ()

[SWS_SoAd_00576][If `SoAd_TpCancelReceive()` or `SoAd_TpCancelTransmit()` is called for a PDU where TP reception or TP transmission is not in progress, SoAd shall ignore the request and return E_NOT_OK.] ()

7.8 Disconnection and recovery

[SWS_SoAd_00577][Within `SoAd_MainFunction()`, SoAd shall close the socket connection for any communication cancelation request related to an active TP transmission.] ()

[SWS_SoAd_00581][Within `SoAd_MainFunction()`, SoAd shall close the socket connection for any communication cancelation request related to an active TP reception.] ()

[SWS_SoAd_00586][SoAd shall automatically reestablish a socket connection which is in the connection state SOAD_SOCON_RECONNECT within `SoAd_MainFunction()` - independent of the configuration parameter SoAdSocketAutomaticSoConSetup, i.e. connection shall be reestablished even if the parameter is set to FALSE. Reconnection shall be done by considering configuration parameter SoAdSocketTcpInitiate.] ()

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

[SWS_SoAd_00694][If a UDP socket connection is configured with a `SoAdSocketUdpAliveSupervisionTimeout` and the remote address was overwritten, as described in [SWS_SoAd_00592] and the alive supervision timer for this socket connection shall be started with the value specified by the configuration parameter `SoAdSocketUdpAliveSupervisionTimeout` and every further reception from the same remote node shall reset the timer back to the initial value at `SoAd_RxIndication()`.] (SRS_Eth_00085)

[SWS_SoAd_00695][If a UDP socket connection is configured with a `SoAdSocketUdpAliveSupervisionTimeout` and the alive supervision timer runs out, SoAd shall

(a) change the state of the socket connection to SOAD_SOCON_RECONNECT,
(b) deactive the alive supervision timer and
(c) reset the remote address to the configured remote address at `SoAd_MainFunction()`.⌋ (SRS_Eth_00085)

7.9 Routing Groups

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

[SWS_SoAd_00601]⌈ SoAd shall maintain the state of each configured routing group and activate or deactivate the state at initialization depending on the configuration parameter SoAdRoutingGroupIsEnabledAtInit.⌋ ()

[SWS_SoAd_00721]⌈ For RoutingGroups that are referenced by a SoAdPduRouteDest that refers to a SocketConnectionGroup SoAd shall maintain independent states for each SocketConnection that is part of the referenced SocketConnectionGroup and handle them as if they were separate RoutingGroups.⌋ ()

[SWS_SoAd_00560]⌈ If `SoAd_IfTransmit()` is called with 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 reception or transmission process by an upper layer (e.g. long TP-PDU which is received via a multiple CopyRxData calls) are not affected.

7.10 PDU fan-out

[SWS_SoAd_00602]⌈ SoAd shall support more than one SoAdPduRouteDest per SoAdPduRoute for upper layers of If-type, i.e. a single IF-PDU can be transmitted via multiple socket connections.⌋ ()

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_00722]⌈ SoAd shall handle SoAdPduRoutes with SoAdPduRouteDests refering to a SocketConnectionGroup as if they were separate SoAdPduRouteDests refering to each SocketConnection of this Group.⌋ ()

[SWS_SoAd_00648]⌈ If a transmit request on any of multiple socket connections returns E_NOT_OK, SoAd shall return E_NOT_OK at `SoAd_IfTransmit()`.⌋ ()

[SWS_SoAd_00647]⌈ In case of multiple socket connections, SoAd shall call the upper layer with the configured transmit confirmation function (`<Up>_[SoAd][If]TxConfirmation>()`) only once after transmission on all related socket connections succeded.⌋ ()

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. ⌋ ( )

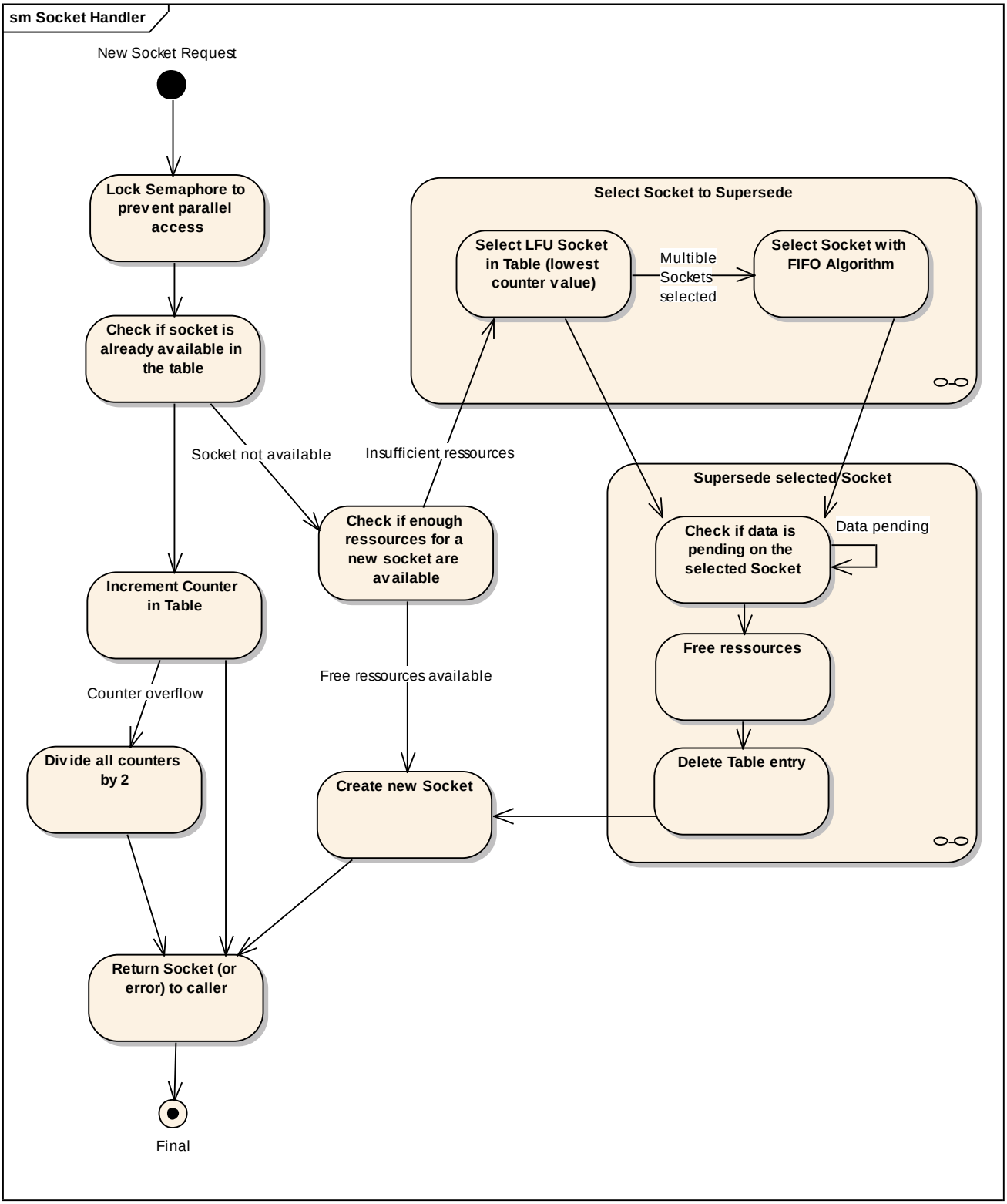


Figure 3: Socket Resource Management

7.12 Buffer handling

[SWS_SoAd_00505][The SoAd shall provide sufficient buffers to store received data which can't be forwarded to the upper layer within the context of SoAd_RxIndication as well as buffers for data which can (or should) not be forwarded to TcpIp.] ()

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

7.13 Error classification

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

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

7.13.1 Development Errors

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

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

] ()

7.13.2 Runtime Errors

< *There are no runtime errors.*>

7.13.3 Transient Faults

< *There are no transient faults.*>

7.13.4 Production Errors

< *There are no production errors.*>

7.13.5 Extended Production Errors

< *There are no extended production errors.*>

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
Std_Types	Std_ReturnType
	Std_VersionInfoType
TcpIp	TcpIp_DomainType
	TcpIp_EventType
	TcpIp_IpAddrAssignmentType
	TcpIp_IpAddrStateType
	TcpIp_LocalAddrIdType
	TcpIp_ParamIdType
	TcpIp_ProtocolType
	TcpIp_ReturnType
	TcpIp_SockAddrType
	TcpIp_SocketIdType
	TcpIp_StateType

⌋ ()

8.2 Type definitions

8.2.1 SoAd_SoConIdType

[SWS_SoAd_00518]⌈

Name:	`SoAd_SoConIdType`		
Type:	`uint8, uint16`		
Range:	`0..<SoAdSoConMax>`	--	Zero-based integer number
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		
Range:	0..<SoAdRoutingGroupMax>	--	Zero-based integer number
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 specific	The content of the configuration data structure is implementation 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:	`void SoAd_Init( const SoAd_ConfigType* SoAdConfigPtr )`
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:	Initializes the Socket Adaptor.

⌋ ( )

[SWS_SoAd_00211] ⌈
`SoAd_Init` shall store the access to the configuration structure for subsequent API calls. ⌋ ( )

[SWS_SoAd_00723]⌈ `SoAd_Init()` initializes all global variables of a Socket Adaptor instance and puts all socket connections into the state `SOAD_SOCON_OFFLINE`.⌋ ()

[SWS_SoAd_00216**]** ⌈
If development error detection is enabled: `SoAd_Init()` shall check the parameter `SoAdConfigPtr` for containing a valid configuration. If the check fails, `SoAd_Init()` shall raise the development error `SOAD_E_INIT_FAILED`. ⌋ ( )

8.3.2 Normal Operation

8.3.2.1 SoAd_IfTransmit

[SWS_SoAd_00091**]** ⌈

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 specifiying the socket to be used for tranmission 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:	Requests transmission of an I-PDU.	

⌋ ( )

[SWS_SoAd_00213**]** ⌈
If development error detection is enabled: `SoAd_IfTransmit()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_IfTransmit()` shall raise the development error `SOAD_E_NOTINIT`.⌋ ( )

[SWS_SoAd_00214**]** ⌈
If development error detection is enabled: `SoAd_IfTransmit()` shall check parameter `SoAdSrcPduId` for being valid. If the check fails, `SoAd_IfTransmit()` shall raise the development error `SOAD_E_INV_PDUID`.⌋ ( )

[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`.] ()

[SWS_SoAd_00658][If development error detection is enabled: `SoAd_IfRoutingGroupTransmit()` shall check parameter `id` for being valid (i.e. id refers a routing group that has configuration parameter SoAdRoutingGroupTxTriggerable set to TRUE). If the check fails, `SoAd_IfRoutingGroupTransmit()` shall raise the development error `SOAD_E_INV_ARG`.] ()

8.3.2.3 SoAd_IfSpecificRoutingGroupTransmit

[SWS_SoAd_00711][

Service name:	SoAd_IfSpecificRoutingGroupTransmit	
Syntax:	`Std_ReturnType SoAd_IfSpecificRoutingGroupTransmit( SoAd_RoutingGroupIdType id, SoAd_SoConIdType SoConId )`	
Service ID[hex]:	0x1f	
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).
	SoConId	socket connection index specifying the socket connection on which the PDUs shall be transmitted
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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 on the socket connection specified by SoConId after requesting the data from the related upper layer.	

] ()

[SWS_SoAd_00712][If development error detection is enabled: `SoAd_IfSpecificRoutingGroupTransmit()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_IfSpecificRoutingGroupTransmit()` shall raise the development error `SOAD_E_NOTINIT`.] ()

[SWS_SoAd_00713][If development error detection is enabled: `SoAd_IfSpecificRoutingGroupTransmit()` shall check parameter `id` for being valid (i.e. id refers a routing group that has configuration parameter SoAdRoutingGroupTxTriggerable set to TRUE). If the check fails, `SoAd_IfSpecificRoutingGroupTransmit()` shall raise the development error `SOAD_E_INV_ARG`.] ()

8.3.2.4 SoAd_TpTransmit

[SWS_SoAd_00105**]** ⌈

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 specifiying the socket to be used for tranmission 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:	Requests transmission of an I-PDU.	

⌋ ( )

[SWS_SoAd_00224**]** ⌈
If development error detection is enabled: `SoAd_TpTransmit()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TpTransmit()` shall raise the development error `SOAD_E_NOTINIT`. ⌋ ( )

[SWS_SoAd_00237**]** ⌈
If development error detection is enabled: `SoAd_TpTransmit()` shall check parameter `SoAdSrcPduId` for being valid. If the check fails, `SoAd_TpTransmit()` shall raise the development error `SOAD_E_INV_PDUID`. ⌋ ( )

[SWS_SoAd_00650]⌈ The service `SoAd_TpTransmit()` shall skip the transmit request and return E_NOT_OK if there is already an IF or TP transmission ongoing on the related socket connection identified by 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]⌈

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	Identifiaction 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`.] ()

[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`.] ()

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	Identifiaction 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.] ()

[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.] ()

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 (inout):	None	
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 the specified TxPduId.	

] ()

[SWS_SoAd_00609][If development error detection is enabled: SoAd_GetSoConId() shall check that the service SoAd_Init was previously called. If the check fails, SoAd_GetSoConId() shall raise the development error SOAD_E_NOTINIT.] ()

[SWS_SoAd_00724][In case SoAd_GetSoConId() is called with a TxPduId related to a SoAdPduRoute with a fan-out (i.e. multiple SoAdPduRouteDest specified), SoAd_GetSoConId() shall skip further processings and return E_NOT_OK.] ()

[SWS_SoAd_00610][If development error detection is enabled: SoAd_GetSoConId() shall check parameter TxPduId for being valid. If the check fails, SoAd_GetSoConId() shall raise the development error SOAD_E_INV_PDUID.] ()

8.3.4.2 SoAd_OpenSoCon

[SWS_SoAd_00510][

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`.] ()

[SWS_SoAd_00611][If development error detection is enabled: `SoAd_OpenSoCon()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_OpenSoCon()` shall raise the development error `SOAD_E_INV_ARG`.] ()

[SWS_SoAd_00528][If development error detection is enabled: In case `SoAd_OpenSoCon()` is called for a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to “TRUE” the development error SOAD_E_INV_ARG shall be raised.] ()

8.3.4.3 SoAd_CloseSoCon

[SWS_SoAd_00511][

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 SoConId.	

] ()

[SWS_SoAd_00616][If development error detection is enabled: `SoAd_CloseSoCon()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_CloseSoCon()` shall raise the development error `SOAD_E_NOTINIT`.] ()

[SWS_SoAd_00612][If development error detection is enabled: `SoAd_CloseSoCon()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_CloseSoCon()` shall raise the development error `SOAD_E_INV_ARG`.] ()

[SWS_SoAd_00529][If development error detection is enabled: In case `SoAd_CloseSoCon()` is called for a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to "TRUE" the development error SOAD_E_INV_ARG shall be raised.] ()

8.3.4.4 SoAd_RequestIpAddrAssignment

[SWS_SoAd_00520][

Service name:	SoAd_RequestIpAddrAssignment	
Syntax:	`Std_ReturnType SoAd_RequestIpAddrAssignment( SoAd_SoConIdType SoConId, TcpIp_IpAddrAssignmentType Type, const TcpIp_SockAddrType* LocalIpAddrPtr, uint8 Netmask, const TcpIp_SockAddrType* DefaultRouterPtr )`	
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 used in case the parameter Type is set to TCPIP_IPADDR_ASSIGNMENT_STATIC, can be set to NULL_PTR otherwise.
	Netmask	Network mask of IPv4 address or address prefix of IPv6 address in CIDR Notation. Note: This parameter is only used in case the parameter Type is set to TCPIP_IPADDR_ASSIGNMENT_STATIC.
	DefaultRouterPtr	Pointer to structure containing the IP address of the default router (gateway) which shall be assigned. Note: This parameter is only used in case the parameter Type is

		set to TCPIP_IPADDR_ASSIGNMENT_STATIC, can be set to NULL_PTR otherwise.
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`.] ()

[SWS_SoAd_00617][If development error detection is enabled, `SoAd_RequestIpAddrAssignment()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_RequestIpAddrAssignment()` shall raise the development error `SOAD_E_INV_ARG`.] ()

8.3.4.5 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`.] ()

[SWS_SoAd_00619][If development error detection is enabled: `SoAd_ReleaseIpAddrAssignment()` shall check parameter `SoConId` for being

valid. If the check fails, `SoAd_ReleaseIpAddrAssignment()` shall raise the development error `SOAD_E_INV_ARG`.⌋ ()

8.3.4.6 SoAd_GetLocalAddr

[SWS_SoAd_00506]⌈

Service name:	SoAd_GetLocalAddr	
Syntax:	`Std_ReturnType SoAd_GetLocalAddr( SoAd_SoConIdType SoConId, TcpIp_SockAddrType* LocalAddrPtr, uint8* NetmaskPtr, TcpIp_SockAddrType* DefaultRouterPtr )`	
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):	LocalAddrPtr	Pointer to a struct where the local address (IP address and port) is stored. The struct member domain shall be set by the caller of the API to the desired TcpIp_DomainType and it shall be ensured by the caller that the struct is large enough to store an address of the selected type (INET or INET6).
	DefaultRouterPtr	Pointer to struct where the IP address of the default router (gateway) is stored (struct member "port" is not used and of arbitrary value). The struct must be of the same type and size as LocalAddrPtr.
Parameters (out):	NetmaskPtr	Pointer to memory where Network mask of IPv4 address or address prefix of IPv6 address in CIDR Notation 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 local address (IP address and port) actually used for the SoAd socket connection specified by SoConId, the netmask and default router	

⌋ ()

[SWS_SoAd_00621]⌈ If development error detection is enabled: `SoAd_GetLocalAddr()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_GetLocalAddr()` shall raise the development error `SOAD_E_NOTINIT`.⌋ ()

[SWS_SoAd_00620]⌈ If development error detection is enabled: `SoAd_GetLocalAddr()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_GetLocalAddr()` shall raise the development error `SOAD_E_INV_ARG`.⌋ ()

8.3.4.7 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`.] ()

[SWS_SoAd_00622][If development error detection is enabled: `SoAd_GetPhysAddr()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_GetPhysAddr()` shall raise the development error `SOAD_E_INV_ARG`.] ()

8.3.4.8 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`.⌋ ()

[SWS_SoAd_00660]⌈ If development error detection is enabled: `SoAd_GetRemoteAddr()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_GetRemoteAddr()` shall raise the development error `SOAD_E_INV_ARG`.⌋ ()

[SWS_SoAd_00666]⌈ `SoAd_GetRemoteAddr()` shall immediately return E_NOT_OK if the remote address of the socket connection specified by parameter SoConId is not set.⌋ ()

[SWS_SoAd_00664]⌈ At `SoAd_GetRemoteAddr()` SoAd shall retrieve the remote address (IP address and port) actually used for the socket connection specified by parameter `SoConId`.⌋ ()

[SWS_SoAd_00698]⌈ `SoAd_GetRemoteAddr()` shall refuse the request if the domain set in `IpAddrPtr` does not match the TcpIp_DomainType of the local address related to the socket connection identified by `SoConId` and return E_NOT_OK. If development error detection is enabled, the service `SoAd_GetRemoteAddr()` shall also raise the development error `SOAD_E_INV_ARG`.⌋ ()

8.3.4.9 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`.] ()

[SWS_SoAd_00625][If development error detection is enabled: `SoAd_EnableRouting()` shall check parameter `id` for being valid. If the check fails, `SoAd_EnableRouting()` shall raise the development error `SOAD_E_INV_ARG`.] ()

8.3.4.10 SoAd_EnableSpecificRouting

[SWS_SoAd_00714][

Service name:	SoAd_EnableSpecificRouting	
Syntax:	`Std_ReturnType SoAd_EnableSpecificRouting( SoAd_RoutingGroupIdType id, SoAd_SoConIdType SoConId )`	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	routing group identifier specifying the routing group to be enabled
	SoConId	socket connection index specifying the socket connection on which the routing group shall be enabled

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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 only on the socket connection identified by SoConId.	

] ()

[SWS_SoAd_00715][If development error detection is enabled: `SoAd_EnableSpecificRouting()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_EnableSpecificRouting()` shall raise the development error `SOAD_E_NOTINIT`.] ()

[SWS_SoAd_00716][If development error detection is enabled: `SoAd_EnableSpecificRouting()` shall check parameter `id` for being valid. If the check fails, `SoAd_EnableSpecificRouting()` shall raise the development error `SOAD_E_INV_ARG`.] ()

8.3.4.11 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`.] ()

[SWS_SoAd_00626][If development error detection is enabled: `SoAd_DisableRouting()` shall check parameter `id` for being valid. If the check

fails, SoAd_DisableRouting() shall raise the development error SOAD_E_INV_ARG.⌋ ()

8.3.4.12 SoAd_DisableSpecificRouting

[SWS_SoAd_00717]⌈

Service name:	SoAd_DisableSpecificRouting	
Syntax:	`Std_ReturnType SoAd_DisableSpecificRouting( SoAd_RoutingGroupIdType id, SoAd_SoConIdType SoConId )`	
Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	routing group identifier specifying the routing group to be disabled
	SoConId	socket connection index specifying the socket connection on which the routing group shall be disabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	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 only on the socket connection identified by SoConId.	

⌋ ()

[SWS_SoAd_00718]⌈ If development error detection is enabled: SoAd_DisableSpecificRouting() shall check that the service SoAd_Init was previously called. If the check fails, SoAd_DisableSpecificRouting() shall raise the development error SOAD_E_NOTINIT.⌋ ()

[SWS_SoAd_00719]⌈ If development error detection is enabled: SoAd_DisableSpecificRouting() shall check parameter id for being valid. If the check fails, SoAd_DisableSpecificRouting() shall raise the development error SOAD_E_INV_ARG.⌋ ()

8.3.4.13 SoAd_SetRemoteAddr

[SWS_SoAd_00515][

Service name:	SoAd_SetRemoteAddr	
Syntax:	`Std_ReturnType SoAd_SetRemoteAddr( SoAd_SoConIdType SoConId, const 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`.] ()

[SWS_SoAd_00531][If development error detection is enabled and SoConId refers to a socket connection with configuration parameter SoAdSocketAutomatic-SoConSetup set to TRUE, the function `SoAd_SetRemoteAddr()` shall raise the development error `SOAD_E_INV_ARG`.] ()

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

(1) a socket connection that is in the mode SOAD_SOCON_OFFLINE or

(2) 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.] ()

[SWS_SoAd_00699][`SoAd_SetRemoteAddr()` shall refuse the request if the domain set in RemoteAddrPtr does not match the TcpIp_DomainType of the local

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

8.3.4.14 SoAd_SetUniqueRemoteAddr

[SWS_SoAd_00671][

Service name:	SoAd_SetUniqueRemoteAddr	
Syntax:	`Std_ReturnType SoAd_SetUniqueRemoteAddr( SoAd_SoConIdType SoConId, const 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.] ()

[SWS_SoAd_00673][If development error detection is enabled: `SoAd_SetUnique-RemoteAddr()` shall check parameter SoConId for being valid. If the check fails, `SoAd_SetUniqueRemoteAddr()` shall raise the development error SOAD_E_INV_ARG] ()

[SWS_SoAd_00675][The function `SoAd_SetUniqueRemoteAddr()` shall check if one of the socket connections of the socket connection group, identified by SoConId, is already configured with the address specified by RemoteAddrPtr. In this case, it shall return the socket connection index via AssignedSoConIdPtr and return E_OK.] ()

[SWS_SoAd_00676][If no socket connection is already configured with the address specified by RemoteAddrPtr, `SoAd_SetUniqueRemoteAddr()` shall:
(1) choose an unused socket connection using the best match algorithm described in [SWS_SoAd_00680]
(2) set it to the remote address specified by RemoteAddrPtr
(3) set AssignedSoConIdPtr to the index of the chosen socket connection and
(4) return E_OK.
A socket connection is “unused” if its actual remote address has an IP address wildcard and/or port wildcard.] ()

[SWS_SoAd_00678][`SoAd_SetUniqueRemoteAddr()` shall reject the request and return E_NOT_OK if there are no unused socket connections within the socket connection group identified by SoConId.] ()

[SWS_SoAd_00700][`SoAd_SetUniqueRemoteAddr()` shall refuse the request if the domain set in `RemoteAddrPtr` does not match the TcpIp_DomainType of the local address related to the socket connection identified by `SoConId` and return E_NOT_OK. If development error detection is enabled, the service `SoAd_SetUniqueRemoteAddr()` shall also raise the development error `SOAD_E_INV_ARG`.] ()

8.3.4.15 SoAd_TpChangeParameter

[SWS_SoAd_00508][

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`.⌋ ()

[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`.⌋ ()

[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`.⌋ ()

8.3.4.16 SoAd_ReadDhcpHostNameOption

[SWS_SoAd_00681] [

Service name:	SoAd_ReadDhcpHostNameOption	
Syntax:	`Std_ReturnType SoAd_ReadDhcpHostNameOption( SoAd_SoConIdType SoConId, uint8* length, uint8* data )`	
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 hostname 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 hostname, i.e. the Fully Qualified Domain Name (FQDN) according to IETF RFC 4702/IETF RFC 4704 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, i.e. FQDN option in the DHCP submodule of the TCP/IP stack.	

] ()

[SWS_SoAd_00701][If development error detection is enabled: `SoAd_ReadDhcpHostNameOption()` shall check that the service SoAd_Init was previously called. If the check fails, `SoAd_ReadDhcpHostNameOption()` shall raise the development error `SOAD_E_NOTINIT`.] ()

[SWS_SoAd_00702][If development error detection is enabled: `SoAd_ReadDhcpHostNameOption()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_ReadDhcpHostNameOption()` shall raise the development error `SOAD_E_INV_ARG`.] ()

[SWS_SoAd_00703][The service `SoAd_ReadDhcpHostNameOption()` shall forward the call to `TcpIp_DhcpReadOption()` with the parameter `Option` set to the option code 81 according to IETF RFC 4702, if the socket connection identified by `SoConId` is related to a local address of the TcpIp_DomainType `TCPIP_AF_INET`.] ()

[SWS_SoAd_00704][The service `SoAd_ReadDhcpHostNameOption()` shall forward the call to `TcpIp_DhcpV6ReadOption()` with the parameter `Option` set to the option code 39 according to IETF RFC 4704, if the socket connection identified by `SoConId` is related to a local address of the TcpIp_DomainType `TCPIP_AF_INET6`.] ()

8.3.4.17 SoAd_WriteDhcpHostNameOption

[SWS_SoAd_00679][

Service name:	SoAd_WriteDhcpHostNameOption	
Syntax:	`Std_ReturnType SoAd_WriteDhcpHostNameOption( SoAd_SoConIdType SoConId, uint8 length, const uint8* data )`	
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 hostname shall be changed
	length	Length of hostname to be set.
	data	Pointer to memory containing the hostname, i.e. the Fully Qualified Domain Name (FQDN) according to IETF RFC 4702/IETF RFC 4704.
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, i.e. FQDN option in the DHCP submodule of the TCP/IP stack.	

] ()

[SWS_SoAd_00705][If development error detection is enabled: `SoAd_WriteDhcpHostNameOption()` shall check that the service SoAd_Init was previously called. If the check fails, `SoAd_WriteDhcpHostNameOption()` shall raise the development error `SOAD_E_NOTINIT`.] ()

[SWS_SoAd_00706][If development error detection is enabled: `SoAd_WriteDhcpHostNameOption()` shall check parameter `SoConId` for being valid. If the check fails, `SoAd_WriteDhcpHostNameOption()` shall raise the development error `SOAD_E_INV_ARG`.] ()

[SWS_SoAd_00707][The service `SoAd_WriteDhcpHostNameOption()` shall forward the call to `TcpIp_DhcpWriteOption()` with the parameter `Option` set to the option code 81 according to IETF RFC 4702, if the socket connection identified by `SoConId` is related to a local address of the TcpIp_DomainType `TCPIP_AF_INET`.] ()

[SWS_SoAd_00708][The service `SoAd_WriteDhcpHostNameOption()` shall forward the call to `TcpIp_DhcpV6WriteOption()` with the parameter `Option` set to the option code 39 according to IETF RFC 4704, if the socket connection identified by `SoConId` is related to a local address of the TcpIp_DomainType `TCPIP_AF_INET6`.] ()

8.4 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:	`void SoAd_RxIndication(`
	`  TcpIp_SocketIdType SocketId,`
	`  const TcpIp_SockAddrType* RemoteAddrPtr,`
	`  uint8* BufPtr,`
	`  uint16 Length`
	`)`	

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`. ⌋ ( )

[SWS_SoAd_00268**]** ⌈
If development error detection is enabled: `SoAd_RxIndication()` shall check parameter `RemoteAddrPtr` for being valid. If the check fails, `SoAd_RxIndication()` shall raise the development error `SOAD_E_INV_ARG`. ⌋ ( )

8.4.2 SoAd_CopyTxData

[SWS_SoAd_00523**]**⌈

Service name:	SoAd_CopyTxData	
Syntax:	`BufReq_ReturnType SoAd_CopyTxData( TcpIp_SocketIdType SocketId, uint8* BufPtr, uint16 BufLength )`	
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	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_NOT_OK: Data has not been copied. Request failed. (No further action for TcpIp required. Later the upper layer might either close the socket or retry the transmit request)
Description:	This service requests to copy data for transmission to the buffer indicated. This call is triggered by TcpIp_Transmit(). Note: The call to <Up>_CopyTxData() may happen in the context of TcpIp_Transmit().	

⌋ ()

[SWS_SoAd_00632]⌈ If development error detection is enabled: `SoAd_CopyTxData()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_CopyTxData()` shall raise the development error `SOAD_E_NOTINIT`.⌋ ()

[SWS_SoAd_00633]⌈ If development error detection is enabled: `SoAd_CopyTxData()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_CopyTxData()` shall raise the development error `SOAD_E_INV_SOCKETID`. ⌋ ()

8.4.3 SoAd_TxConfirmation

[SWS_SoAd_00098] ⌈

Service name:	SoAd_TxConfirmation	

Syntax:	`void SoAd_TxConfirmation(`
	`  TcpIp_SocketIdType SocketId,`
	`  uint16 Length`
	`)`	

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:	The TCP/IP stack calls this function after the data has been acknowledged by the peer for TCP. Caveats: The upper layer might not be able to determine exactly which data bytes have been confirmed.	

⌋ ( )

[SWS_SoAd_00269] ⌈
If development error detection is enabled: `SoAd_TxConfirmation()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TxConfirmation()` shall raise the development error `SOAD_E_NOTINIT`. ⌋ ( )

[SWS_SoAd_00270] ⌈
If development error detection is enabled: `SoAd_TxConfirmation()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TxConfirmation()` shall raise the development error `SOAD_E_INV_SOCKETID`. ⌋ ( )

[SWS_SoAd_00271] ⌈
If development error detection is enabled: `SoAd_TxConfirmation()` shall check parameter `Length` for being valid. If the check fails, `SoAd_TxConfirmation()` shall raise the development error `SOAD_E_INV_ARG`. ⌋ ( )

8.4.4 SoAd_TcpAccepted

[SWS_SoAd_00099] ⌈

Service name:	SoAd_TcpAccepted	

Syntax:	`Std_ReturnType SoAd_TcpAccepted(`
	`  TcpIp_SocketIdType SocketId,`
	`  TcpIp_SocketIdType SocketIdConnected,`
	`  const 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 TcpIp_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 upper layer accepts the established connection E_NOT_OK upper layer refuses the established connection, TcpIp stack shall close the connection.
Description:	This service gets called if the stack put a socket into the listen mode before (as server) and a peer connected to it (as client). In detail: The TCP/IP stack calls this function after a socket was set into the listen state with TcpIp_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`. ⌋ ( )

8.4.5 SoAd_TcpConnected

[SWS_SoAd_00100] ⌈

Service name:	SoAd_TcpConnected	
Syntax:	`void SoAd_TcpConnected( TcpIp_SocketIdType SocketId )`	
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:	This service gets called if the stack initiated a TCP connection before (as client) and the peer (the server) acknowledged the connection set up. In detail: The TCP/IP stack calls this function after a socket was requested to connect with TcpIp_TcpConnect() and a TCP connection is confirmed by the peer. The parameter value of SocketId equals the SocketId value of the preceeding TcpIp_TcpConnect() call.	

⌋ ( )

[SWS_SoAd_00274] ⌈
If development error detection is enabled: `SoAd_TcpConnected()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TcpConnected()` shall raise the development error `SOAD_E_NOTINIT`. ⌋ ( )

[SWS_SoAd_00275] ⌈
If development error detection is enabled: `SoAd_TcpConnected()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TcpConnected()` shall raise the development error `SOAD_E_INV_SOCKETID`. ⌋ ( )

8.4.6 SoAd_TcpIpEvent

[SWS_SoAd_00146] ⌈

Service name:	SoAd_TcpIpEvent	
Syntax:	`void SoAd_TcpIpEvent( TcpIp_SocketIdType SocketId, TcpIp_EventType Event )`	
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:	This service gets called if the stack encounters a condition described by the values in Event.

⌋ ( )

[SWS_SoAd_00276]⌈
If development error detection is enabled: `SoAd_TcpIpEvent()` shall check that the service `SoAd_Init` was previously called. If the check fails, `SoAd_TcpIpEvent()` shall raise the development error `SOAD_E_NOTINIT`. ⌋ ( )

[SWS_SoAd_00277]⌈
If development error detection is enabled: `SoAd_TcpIpEvent()` shall check parameter `SocketId` for being valid. If the check fails, `SoAd_TcpIpEvent()` shall raise the development error `SOAD_E_INV_SOCKETID`. ⌋ ( )

[SWS_SoAd_00278]⌈
If development error detection is enabled: `SoAd_TcpIpEvent()` shall check parameter `Event` for being valid. If the check fails, `SoAd_TcpIpEvent()` shall raise the development error `SOAD_E_INV_ARG`. ⌋ ( )

8.4.7 SoAd_LocalIpAddrAssignmentChg

[SWS_SoAd_00209] ⌈

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:	This service gets called by the TCP/IP stack if an IP address assignment changes (i.e. new address assigned or assigned address becomes invalid).	

⌋ ( )

[SWS_SoAd_00279] ⌈
If development error detection is enabled: `SoAd_LocalIpAddrAssignmentChg()` shall check that the service `SoAd_Init` was previously called. If the check fails,

SoAd_LocalIpAddrAssignmentChg() shall raise the development error SOAD_E_NOTINIT. ⌋ ( )

[SWS_SoAd_00280**]** ⌈
If development error detection is enabled: SoAd_LocalIpAddrAssignmentChg() shall check parameter IpAddrId for being valid. If the check fails, SoAd_LocalIpAddrAssignmentChg() shall raise the development error SOAD_E_INV_ARG. ⌋ ( )

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() shall be determined by configuration parameter SOAD_MAINFUNCTION_PERIOD. ⌋ ( )

[SWS_SoAd_00283**]** ⌈
If development error detection is enabled: SoAd_MainFunction() shall check that the service SoAd_Init was previously called. If the check fails, SoAd_MainFunction() shall raise the development error SOAD_E_NOTINIT. ⌋ ( )

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

[SWS_SoAd_00504][

API function	*Description*
TcpIp_Bind	By this API service the TCP/IP stack is requested to bind a UDP or TCP socket to a local resource.
TcpIp_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.
TcpIp_Close	By this API service the TCP/IP stack is requested to close the socket and release all related resources.
TcpIp_GetCtrlIdx	TcpIp_GetCtrlIdx returns the index of the controller related to LocalAddrId.
TcpIp_GetIpAddr	Obtains the local IP address actually used by LocalAddrId, the netmask and default router
TcpIp_GetPhysAddr	Obtains the physical source address used by the EthIf controller implicitly specified via LocalAddrId.
TcpIp_GetRemotePhysAddr	TcpIp_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.
TcpIp_ReleaseIpAddrAssignment	By this API service the local IP address assignment for the IP address specified by LocalAddrId shall be released.
TcpIp_RequestComMode	By this API service the TCP/IP stack is requested to change the TcpIp state of the communication network identified by EthIf controller index.
TcpIp_RequestIpAddrAssignment	By this API service the local IP address assignment for the IP address specified by LocalAddrId shall be initiated.
TcpIp_SoAdGetSocket	By this API service the TCP/IP stack is requested to allocate a new socket. Note: Each accepted incoming TCP connection also allocates a socket resource.
TcpIp_TcpConnect	By this API service the TCP/IP stack is requested to establish a TCP connection to the configured peer.
TcpIp_TcpListen	By this API service the TCP/IP stack is requested to listen on the TCP socket specified by the socket identifier.
TcpIp_TcpReceived	By this API service the reception of socket data is confirmed to the TCP/IP stack.
TcpIp_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) .
TcpIp_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.
Parameters (inout):	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
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:	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	

⌋()

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 PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	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 id,`
	`  const PduInfoType* info,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the received I-PDU.
	info	Provides the source buffer (SduDataPtr) and the number of bytes to be copied (SduLength). An SduLength of 0 can be used to query the current amount

		of available buffer in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
Parameters (inout):	None	
Parameters (out):	bufferSizePtr	Available receive buffer after data has been copied.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data copied successfully BUFREQ_E_NOT_OK: Data was not copied because an error occurred.
Description:	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining data is written to the position indicated by bufferSizePtr.	

⌋ ( )

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 occured.	

⌋ ( )

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 id,`
	`  const PduInfoType* info,`
	`  RetryInfoType* retry,`
	`  PduLengthType* availableDataPtr`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	id	Identification of the transmitted I-PDU.
	info	Provides the destination buffer (SduDataPtr) and the number of bytes to be copied (SduLength). If not enough transmit data is available, no data is copied by the upper layer module and BUFREQ_E_BUSY is returned.

		The lower layer module may retry the call. An SduLength of 0 can be used to indicate state changes in the retry parameter or to query the current amount of available data in the upper layer module. In this case, the SduDataPtr may be a NULL_PTR.
	retry	This parameter is used to acknowledge transmitted data or to retransmit data after transmission problems. If the retry parameter is a NULL_PTR, it indicates that the transmit data can be removed from the buffer immediately after it has been copied. Otherwise, the retry parameter must point to a valid RetryInfoType element. If TpDataState indicates TP_CONFPENDING, the previously copied data must remain in the TP buffer to be available for error recovery. TP_DATACONF indicates that all data that has been copied before this call is confirmed and can be removed from the TP buffer. Data copied by this API call is excluded and will be confirmed later. TP_DATARETRY indicates that this API call shall copy previously copied data in order to recover from an error. In this case TxTpDataCnt specifies the offset in bytes from the current data copy position.
Parameters (inout):	None	
Parameters (out):	availableDataPtr	Indicates the remaining number of bytes that are available in the upper layer module's Tx buffer. availableDataPtr can be used by TP modules that support dynamic payload lengths (e.g. FrIsoTp) to determine the size of the following CFs.
Return value:	BufReq_ReturnType	BUFREQ_OK: Data has been copied to the transmit buffer completely as requested. BUFREQ_E_BUSY: Request could not be fulfilled, because the required amount of Tx data is not available. The lower layer module may retry this call later on. No data has been copied. BUFREQ_E_NOT_OK: Data has not been copied. Request failed.
Description:	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.	

⌋ ( )

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

[SWS_SoAd_00181**]** ⌈

Service name:	<Up>_[SoAd][Tp]TxConfirmation	

Syntax:	`void <Up>_[SoAd][Tp]TxConfirmation(`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`	

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	TxPduId	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:	`void <Up>_SoConModeChg(`
	`  SoAd_SoConIdType SoConId,`
	`  SoAd_SoConModeType Mode`
	`)`	

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

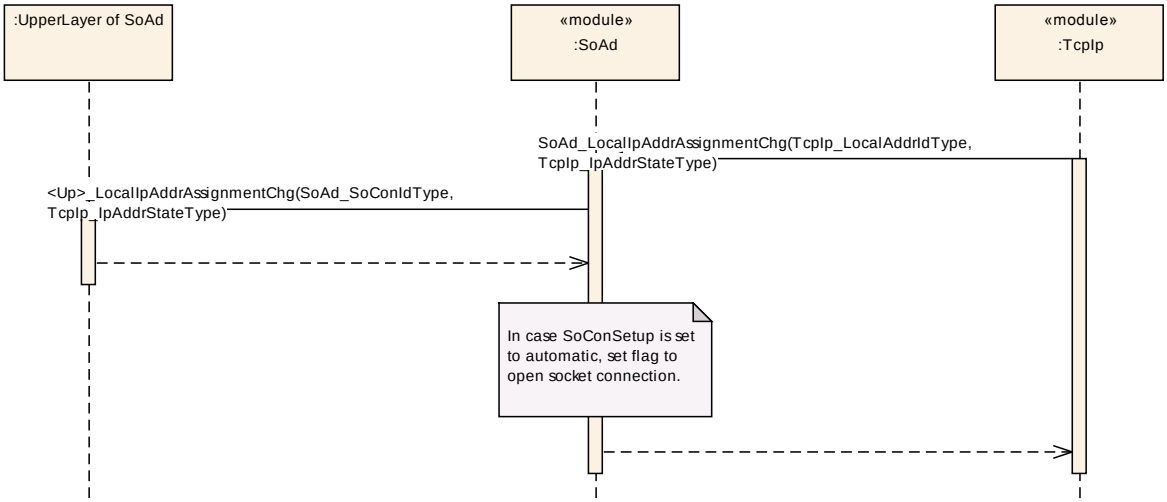
[SWS_SoAd_00513][

Service name:	<Up>_LocalIpAddrAssignmentChg	
Syntax:	`void <Up>_LocalIpAddrAssignmentChg( SoAd_SoConIdType SoConId, TcpIp_IpAddrStateType State )`	
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 assigment 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).	

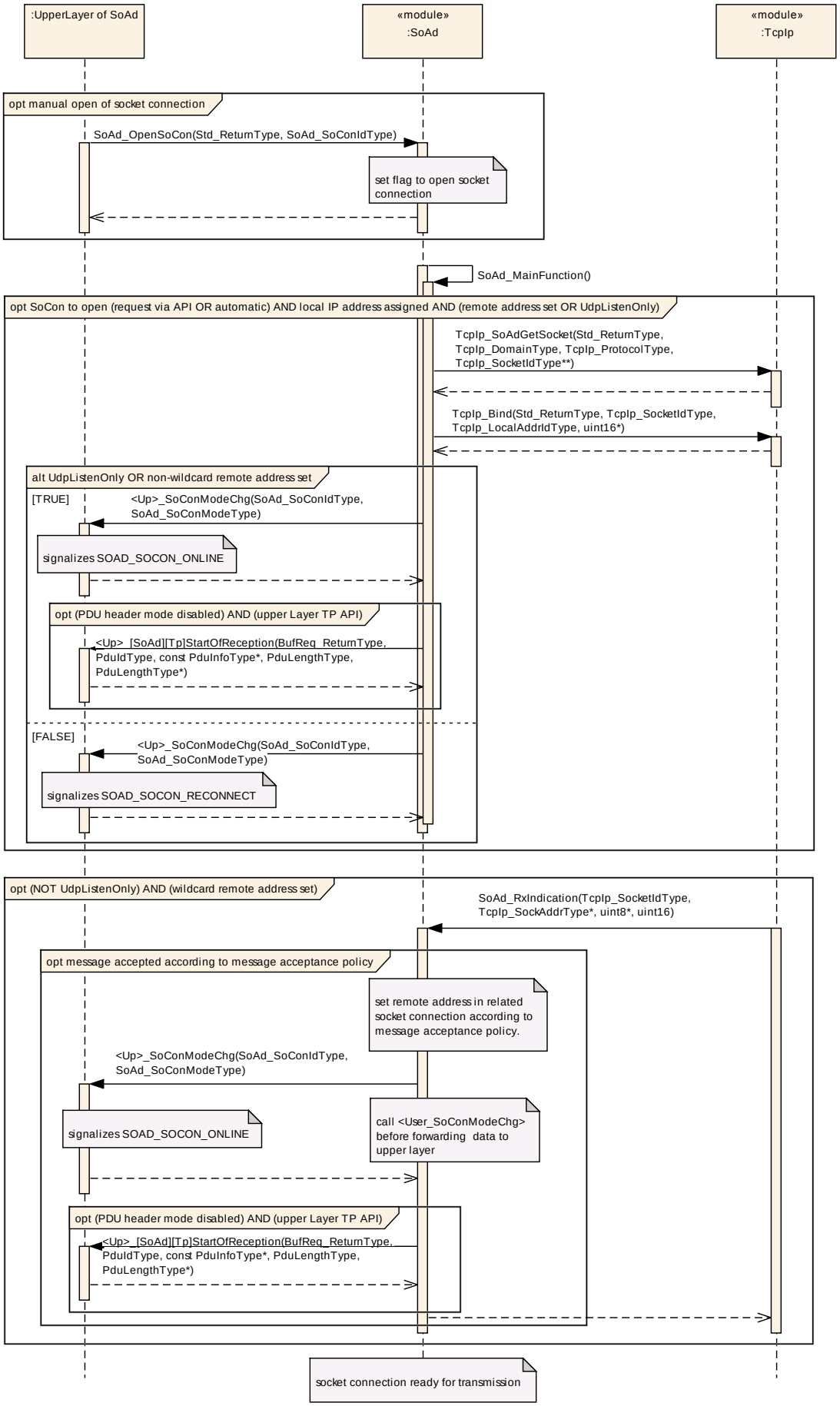
] ()

9 Sequence diagrams and Transition Tables

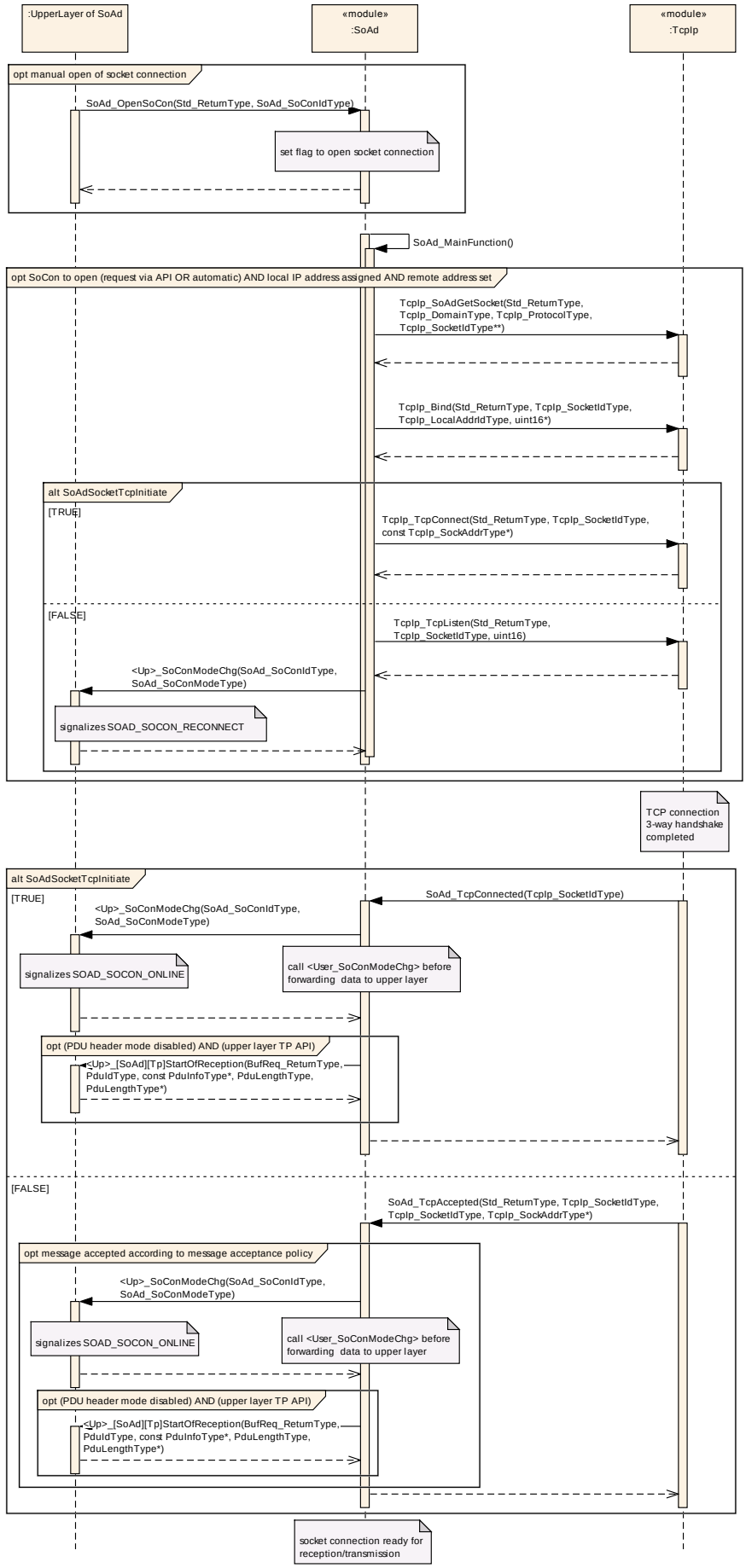
9.1 Address – Assignment



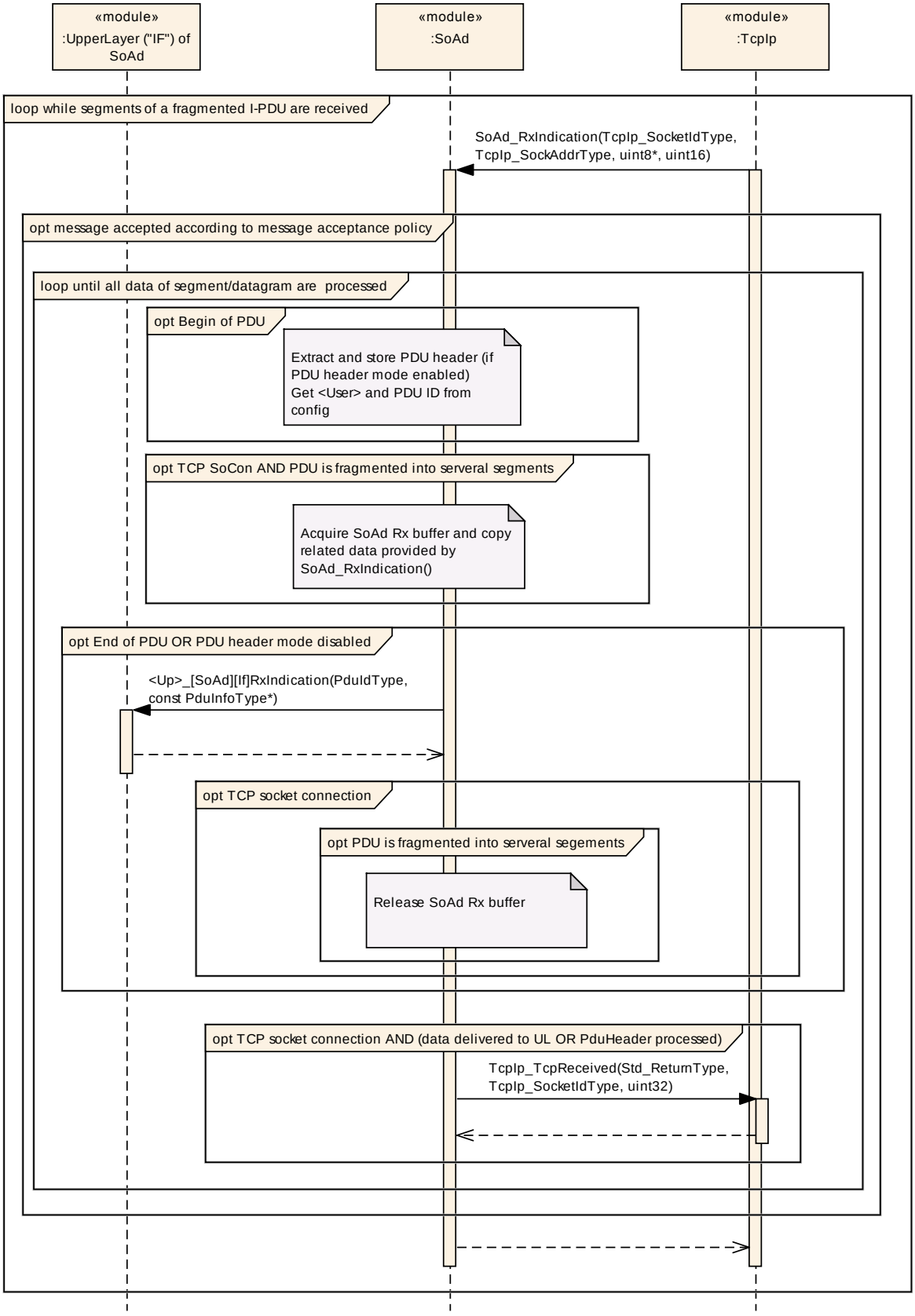
9.2 Socket Connection Setup – UDP


:UpperLayer of SoAd
«module»
:SoAd
«module»
:TcpIp
opt manual open of socket connection
SoAd_OpenSoCon(Std_ReturnType, SoAd_SoConIdType)
set flag to open socket connection
SoAd_MainFunction()
opt SoCon to open (request via API OR automatic) AND local IP address assigned AND (remote address set OR UdpListenOnly)
TcpIp_SoAdGetSocket(Std_ReturnType, TcpIp_DomainType, TcpIp_ProtocolType, TcpIp_SocketIdType**)
TcpIp_Bind(Std_ReturnType, TcpIp_SocketIdType, TcpIp_LocalAddrIdType, uint16*)
alt UdpListenOnly OR non-wildcard remote address set
[TRUE]
<Up>_SoConModeChg(SoAd_SoConIdType, SoAd_SoConModeType)
signalizes SOAD_SOCON_ONLINE
opt (PDU header mode disabled) AND (upper Layer TP API)
<Up>_[SoAd][Tp]StartOfReception(BufReq_ReturnType, PduIdType, const PduInfoType*, PduLengthType, PduLengthType*)
[FALSE]
<Up>_SoConModeChg(SoAd_SoConIdType, SoAd_SoConModeType)
signalizes SOAD_SOCON_RECONNECT
opt (NOT UdpListenOnly) AND (wildcard remote address set)
SoAd_RxIndication(TcpIp_SocketIdType, TcpIp_SockAddrType*, uint8*, uint16)
opt message accepted according to message acceptance policy
set remote address in related socket connection according to message acceptance policy.
<Up>_SoConModeChg(SoAd_SoConIdType, SoAd_SoConModeType)
signalizes SOAD_SOCON_ONLINE
call <User_SoConModeChg> before forwarding data to upper layer
opt (PDU header mode disabled) AND (upper Layer TP API)
<Up>_[SoAd][Tp]StartOfReception(BufReq_ReturnType, PduIdType, const PduInfoType*, PduLengthType, PduLengthType*)
socket connection ready for transmission

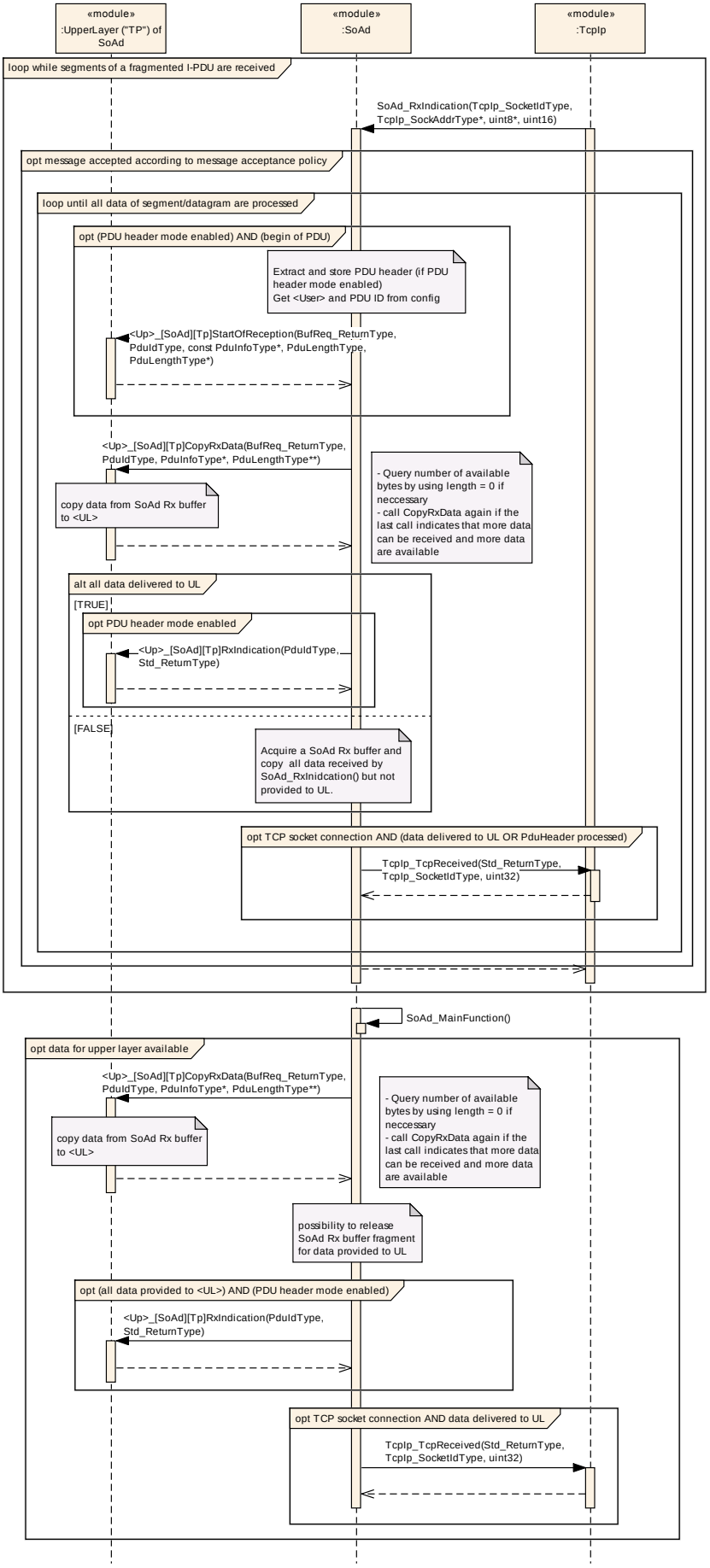
9.3 Socket Connection Setup – TCP


:UpperLayer of SoAd
«module»
:SoAd
«module»
:TcpIp
opt manual open of socket connection
SoAd_OpenSoCon(Std_ReturnType, SoAd_SoConIdType)
set flag to open socket connection
SoAd_MainFunction()
opt SoCon to open (request via API OR automatic) AND local IP address assigned AND remote address set
TcpIp_SoAdGetSocket(Std_ReturnType, TcpIp_DomainType, TcpIp_ProtocolType, TcpIp_SocketIdType**)
TcpIp_Bind(Std_ReturnType, TcpIp_SocketIdType, TcpIp_LocalAddrIdType, uint16*)
alt SoAdSocketTcpInitiate
[TRUE]
TcpIp_TcpConnect(Std_ReturnType, TcpIp_SocketIdType, const TcpIp_SockAddrType*)
[FALSE]
TcpIp_TcpListen(Std_ReturnType, TcpIp_SocketIdType, uint16)
<Up>_SoConModeChg(SoAd_SoConIdType, SoAd_SoConModeType)
signalizes SOAD_SOCON_RECONNECT
TCP connection 3-way handshake completed
alt SoAdSocketTcpInitiate
[TRUE]
SoAd_TcpConnected(TcpIp_SocketIdType)
<Up>_SoConModeChg(SoAd_SoConIdType, SoAd_SoConModeType)
signalizes SOAD_SOCON_ONLINE
call <User_SoConModeChg> before forwarding data to upper layer
opt (PDU header mode disabled) AND (upper layer TP API)
<Up>_[SoAd][Tp]StartOfReception(BufReq_ReturnType, PduIdType, const PduInfoType*, PduLengthType, PduLengthType*)
[FALSE]
SoAd_TcpAccepted(Std_ReturnType, TcpIp_SocketIdType, TcpIp_SocketIdType, TcpIp_SockAddrType*)
opt message accepted according to message acceptance policy
<Up>_SoConModeChg(SoAd_SoConIdType, SoAd_SoConModeType)
signalizes SOAD_SOCON_ONLINE
call <User_SoConModeChg> before forwarding data to upper layer
opt (PDU header mode disabled) AND (upper layer TP API)
<Up>_[SoAd][Tp]StartOfReception(BufReq_ReturnType, PduIdType, const PduInfoType*, PduLengthType, PduLengthType*)
socket connection ready for reception/transmission

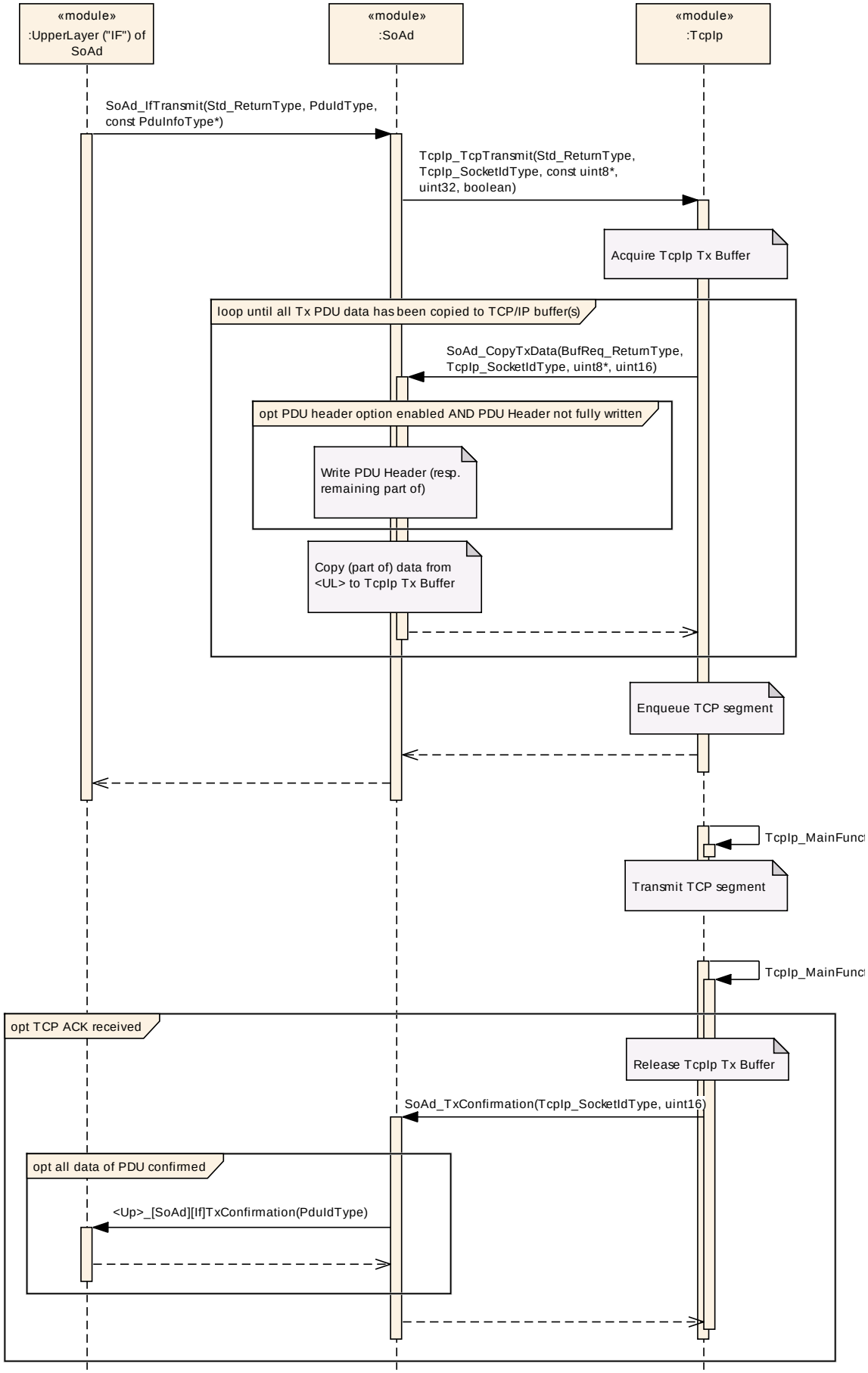
9.4 Reception – Upper Layer If API



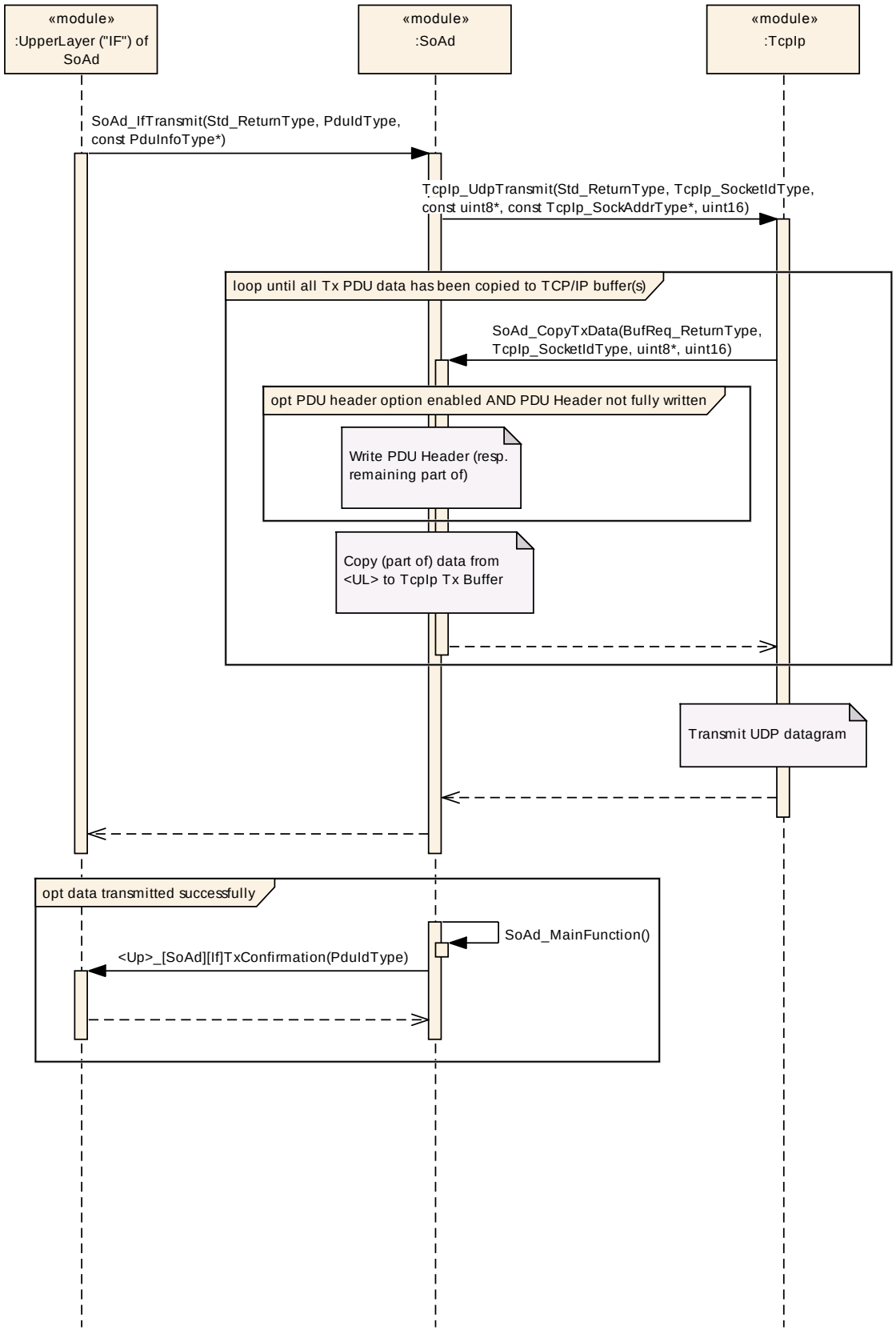
9.5 Reception – Upper Layer TP API


«module»
:UpperLayer ("TP") of SoAd
«module»
:SoAd
«module»
:TcpIp
loop while segments of a fragmented I-PDU are received
SoAd_RxIndication(TcpIp_SocketIdType, TcpIp_SockAddrType*, uint8*, uint16)
opt message accepted according to message acceptance policy
loop until all data of segment/datagram are processed
opt (PDU header mode enabled) AND (begin of PDU)
Extract and store PDU header (if PDU header mode enabled)
Get <User> and PDU ID from config
<Up>_[SoAd][Tp]StartOfReception(BufReq_ReturnType, PduIdType, const PduInfoType*, PduLengthType, PduLengthType*)
<Up>_[SoAd][Tp]CopyRxData(BufReq_ReturnType, PduIdType, PduInfoType*, PduLengthType**)
copy data from SoAd Rx buffer to <UL>
- Query number of available bytes by using length = 0 if neccessary
- call CopyRxData again if the last call indicates that more data can be received and more data are available
alt all data delivered to UL
[TRUE]
opt PDU header mode enabled
<Up>_[SoAd][Tp]RxIndication(PduIdType, Std_ReturnType)
[FALSE]
Acquire a SoAd Rx buffer and copy all data received by SoAd_RxInidcation() but not provided to UL.
opt TCP socket connection AND (data delivered to UL OR PduHeader processed)
TcpIp_TcpReceived(Std_ReturnType, TcpIp_SocketIdType, uint32)
SoAd_MainFunction()
opt data for upper layer available
<Up>_[SoAd][Tp]CopyRxData(BufReq_ReturnType, PduIdType, PduInfoType*, PduLengthType**)
copy data from SoAd Rx buffer to <UL>
- Query number of available bytes by using length = 0 if neccessary
- call CopyRxData again if the last call indicates that more data can be received and more data are available
possibility to release SoAd Rx buffer fragment for data provided to UL
opt (all data provided to <UL>) AND (PDU header mode enabled)
<Up>_[SoAd][Tp]RxIndication(PduIdType, Std_ReturnType)
opt TCP socket connection AND data delivered to UL
TcpIp_TcpReceived(Std_ReturnType, TcpIp_SocketIdType, uint32)

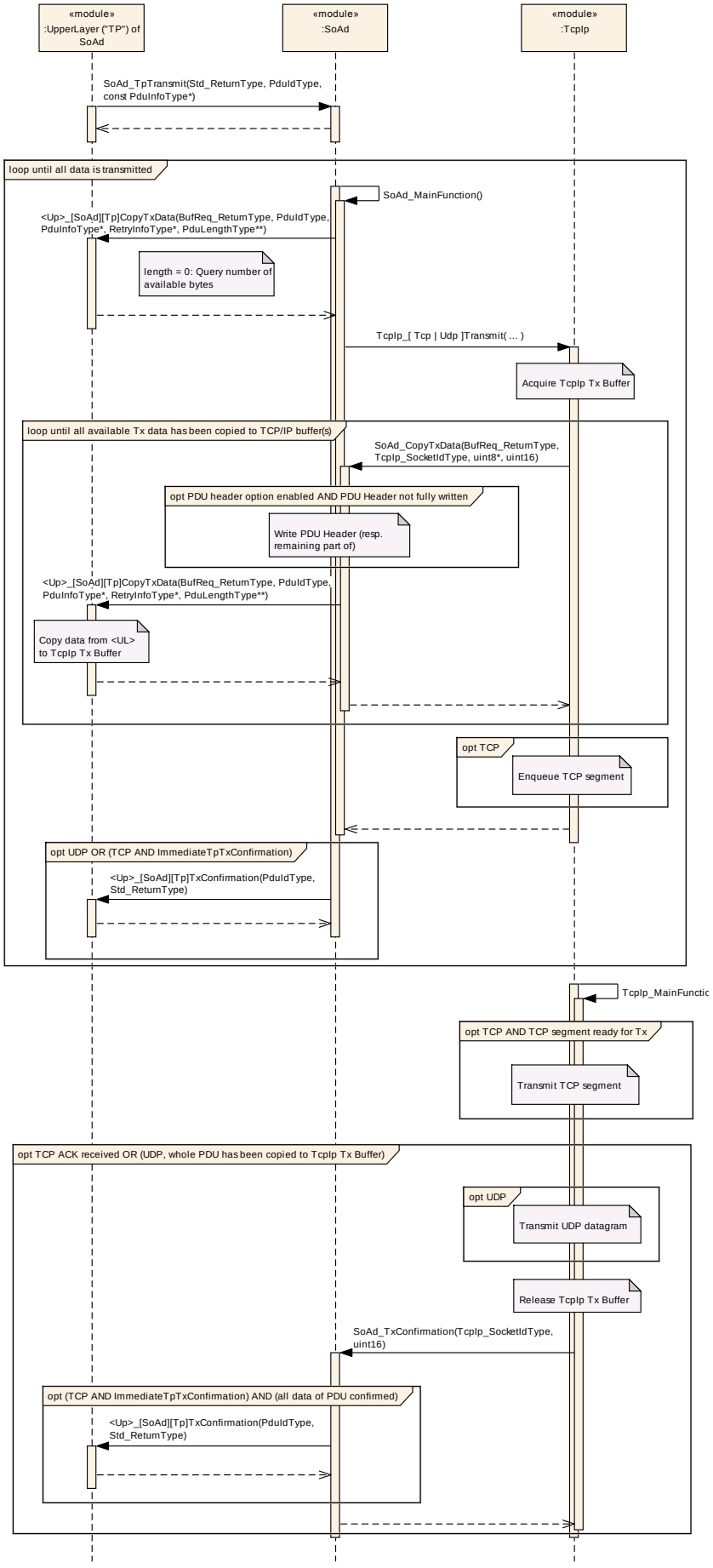
9.6 Transmission – Upper Layer If API – TCP


«module»
:UpperLayer ("IF") of SoAd
«module»
:SoAd
«module»
:TcpIp
SoAd_IfTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
TcpIp_TcpTransmit(Std_ReturnType, TcpIp_SocketIdType, const uint8*, uint32, boolean)
Acquire TcpIp Tx Buffer
loop until all Tx PDU data has been copied to TCP/IP buffer(s)
SoAd_CopyTxData(BufReq_ReturnType, TcpIp_SocketIdType, uint8*, uint16)
opt PDU header option enabled AND PDU Header not fully written
Write PDU Header (resp. remaining part of)
Copy (part of) data from <UL> to TcpIp Tx Buffer
Enqueue TCP segment
TcpIp_MainFunc
Transmit TCP segment
TcpIp_MainFunc
opt TCP ACK received
Release TcpIp Tx Buffer
SoAd_TxConfirmation(TcpIp_SocketIdType, uint16)
opt all data of PDU confirmed
<Up>_[SoAd][If]TxConfirmation(PduIdType)

9.7 Transmission – Upper Layer If API – UDP



9.8 Transmission – Upper Layer TP API


«module»
:UpperLayer ("TP") of SoAd
«module»
:SoAd
«module»
:TcpIp
SoAd_TpTransmit(Std_ReturnType, PduIdType, const PduInfoType*)
loop until all data is transmitted
SoAd_MainFunction()
<Up>_[SoAd][Tp]CopyTxData(BufReq_ReturnType, PduIdType, PduInfoType*, RetryInfoType*, PduLengthType**)
length = 0: Query number of available bytes
TcpIp_[Tcp | Udp]Transmit(...)
Acquire TcpIp Tx Buffer
loop until all available Tx data has been copied to TCP/IP buffer(s)
SoAd_CopyTxData(BufReq_ReturnType, TcpIp_SocketIdType, uint8*, uint16)
opt PDU header option enabled AND PDU Header not fully written
Write PDU Header (resp. remaining part of)
<Up>_[SoAd][Tp]CopyTxData(BufReq_ReturnType, PduIdType, PduInfoType*, RetryInfoType*, PduLengthType**)
Copy data from <UL> to TcpIp Tx Buffer
opt TCP
Enqueue TCP segment
opt UDP OR (TCP AND ImmediateTpTxConfirmation)
<Up>_[SoAd][Tp]TxConfirmation(PduIdType, Std_ReturnType)
TcpIp_MainFunctic
opt TCP AND TCP segment ready for Tx
Transmit TCP segment
opt TCP ACK received OR (UDP, whole PDU has been copied to TcpIp Tx Buffer)
opt UDP
Transmit UDP datagram
Release TcpIp Tx Buffer
SoAd_TxConfirmation(TcpIp_SocketIdType, uint16)
opt (TCP AND ImmediateTpTxConfirmation) AND (all data of PDU confirmed)
<Up>_[SoAd][Tp]TxConfirmation(PduIdType, Std_ReturnType)

10 Configuration specification

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

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

Chapter 10.3 specifies published information of module SoAd.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

The configuration parameters as defined in this chapter are used to create a data model for an AUTOSAR tool chain. The realization in the code is implementation specific.
The configuration parameters are divided into parameters used to enable features, parameters affecting all instances of the UdpNm and parameters affecting the respective instances of the UdpNm.

[SWS_SoAd_00001**]** ⌈All configuration items shall be located outside the kernel of the module.⌋ ( )

[SWS_SoAd_00208**]** ⌈All timing parameters given as EcucFloatParamDef in unit seconds in the configuration, shall be converted to integer multiples of the parameter SoAdMainFunctionPeriod.⌋ ( )

10.2.1 Variants

[SWS_SoAd_00725]⌈ VARIANT-PRE-COMPILE: All configuration parameters shall be configurable at pre-compile time.
Use case: Source code optimization. ⌋ (*SRS_BSW_00345*)

[SWS_SoAd_00726]⌈ VARIANT-LINK-TIME: All configuration parameters of the container SoAdGeneral 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. ⌋ (*SRS_BSW_00344*)

[SWS_SoAd_00727]⌈ VARIANT-POST-BUILD: Most parameters contained in SoAdConfig are configurable at post-build time. The parameters contained in SoAdGeneral are configurable at pre-compile time
Use case: ECU configuration can be flashed (L) and selected during initialization phase (M). ⌋ (*SRS_BSW_00404, SRS_BSW_00405*)

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.
Post-Build Variant Support	true

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.
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		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00145 :**		
Name	SoAdIfTriggerTransmit		
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		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_SoAd_00106 :**		
Name	SoAdIfTxConfirmation		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

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

SWS Item	**ECUC_SoAd_00107 :**		
Name	SoAdSoConModeChg		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00105 :**		
Name	SoAdTp		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_SoAd_00128 :**		
Name	SoAdUseCallerInfix		
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		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00129 :**		
Name	SoAdUseTypeInfix		
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		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00124 :**		
Name	SoAdBswModuleRef		
Description	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. SoAd has to figure out from the structure of the referenced BSW module's configuration, what kind of upper layer he deals with. In case of a CDD SoAd expects UL-APIs in form of _SoAd<If|Tp><function> 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.		
Multiplicity	1		
Type	Foreign reference to [ECUC-MODULE-CONFIGURATION-VALUES]		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

No Included Containers

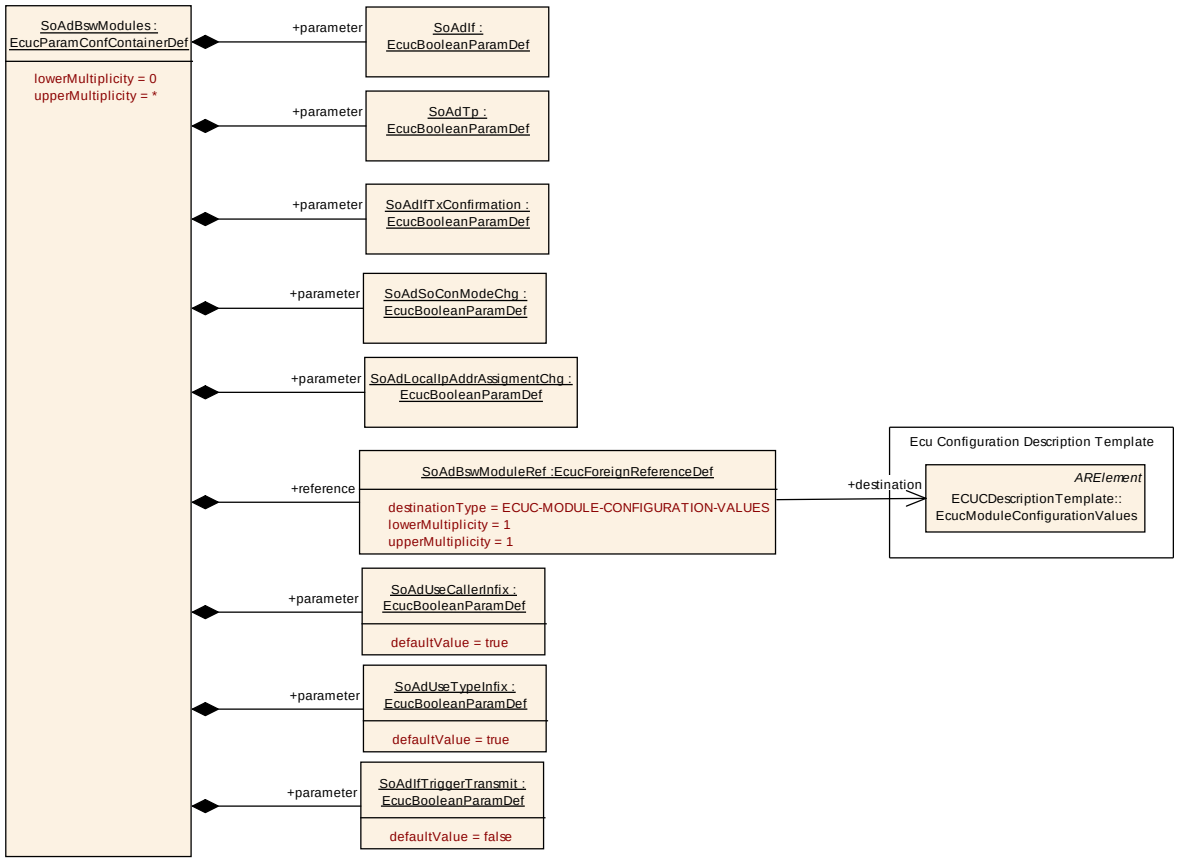


Figure 4: SoAd BswModules container

10.2.4 SoAdGeneral

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

SWS Item	**ECUC_SoAd_00002 :**		
Name	SoAdDevErrorDetect		
Description	Switches the Default Error Tracer (Det) detection and notification ON or OFF. • true: enabled (ON). • false: disabled (OFF).		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00039 :**		
Name	SoAdIPv6AddressEnabled		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

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

Scope / Dependency	scope: ECU

SWS Item	**ECUC_SoAd_00127 :**		
Name	SoAdRoutingGroupMax		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00126 :**		
Name	SoAdSoConMax		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00004 :**		
Name	SoAdVersionInfoApi		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***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
Description	This container contains the configuration parameters and sub containers of the AUTOSAR SoAd module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
SoAdPduRoute	0..*	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission.
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 transmited via a TCP or UDP socket.
SoAdSocketRoute	0..*	Describes the path of a PDU from a socket in the TCP/IP stack to an upper layer of the SoAd after reception in the TCP/IP Stack.

10.2.6 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 transmited via a TCP or UDP socket.
Configuration Parameters	

SWS Item	**ECUC_SoAd_00131 :**		
Name	SoAdPduHeaderEnable		
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	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local

SWS Item	**ECUC_SoAd_00067 :**		
Name	SoAdResourceManagementEnable		
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	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00110 :**		
Name	SoAdSocketAutomaticSoConSetup		
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		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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

Scope / Dependency	scope: local

SWS Item	**ECUC_SoAd_00112 :**		
Name	SoAdSocketIpAddrAssignmentChgNotification		
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		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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

SWS Item	**ECUC_SoAd_00137 :**		
Name	SoAdSocketMsgAcceptanceFilterEnabled		
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		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: SoAdSocketRemoteAddress, SoAdSocketUdpListenOnly, SoAdSocketConnectionGroup

SWS Item	**ECUC_SoAd_00111 :**		
Name	SoAdSocketSoConModeChgNotification		
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		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00134 :**		
Name	SoAdSocketTpRxBufferMin		
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	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

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

Scope / Dependency	scope: 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		
Description	Socket connection identifier used as SoConId in the interaction with upper layers.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
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_00149 :**		
Name	SoAdSocketUdpAliveSupervisionTimeout		
Description	Specifies the time in [s] a UDP socket connection remains in the mode SOAD_SOCON_ONLINE after the latest reception of a frame from the remote peer specified by the remote address. If this optional parameter is not enabled UDP Alive Supervision is deactivated for the related socket connection group.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local

SWS Item	**ECUC_SoAd_00024 :**		
Name	SoAdSocketUdpListenOnly		
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		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: SoAdSocketProtocol		

SWS Item	**ECUC_SoAd_00154 :**		
Name	SoAdSocketUdpStrictHeaderLenCheckEnabled		
Description	Specifies if UDP messages shall be dropped (TRUE) if the length of all contained PDUs does not match the length of the whole message or not (FALSE). Shall only be set to TRUE if SoAdPduHeaderEnable is also set to TRUE.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: SoAdPduHeaderEnable		

SWS Item	**ECUC_SoAd_00133 :**		
Name	SoAdSocketUdpTriggerTimeout		
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	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: SoAdTxUdpTriggerMode

SWS Item	**ECUC_SoAd_00135 :**		
Name	SoAdSocketnPduUdpTxBufferMin		
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 is configured, the required buffer size can be determined automatically. This optional parameter is only relevant if a nPduUdpTxBuffer is used.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local 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	SoAdSocketTcpImmediateTpTxConfirmation		
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 TcpIp.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: ECU

SWS Item	**ECUC_SoAd_00022 :**		
Name	SoAdSocketTcpInitiate		
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	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00148 :**		
Name	SoAdSocketTcpKeepAlive		
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		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: TcpIpTcpKeepAliveEnabled		

SWS Item	**ECUC_SoAd_00152 :**		
Name	SoAdSocketTcpKeepAliveInterval		
Description	Specifies the interval in seconds between subsequent keepalive probes.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: SoAdSocketTcpKeepAlive

SWS Item	**ECUC_SoAd_00151 :**		
Name	SoAdSocketTcpKeepAliveProbesMax		
Description	Maximum number of times that TCP retransmits an individual data segment before aborting the connection.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: SoAdSocketTcpKeepAlive		

SWS Item	**ECUC_SoAd_00153 :**		
Name	SoAdSocketTcpKeepAliveTime		
Description	Specifies the time in seconds between the last data packet sent and the first keepalive probe.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	7200		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: SoAdSocketTcpKeepAlive

SWS Item	**ECUC_SoAd_00023 :**		
Name	SoAdSocketTcpNoDelay		
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	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: TcpIpTcpNagleEnabled		

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

No Included Containers

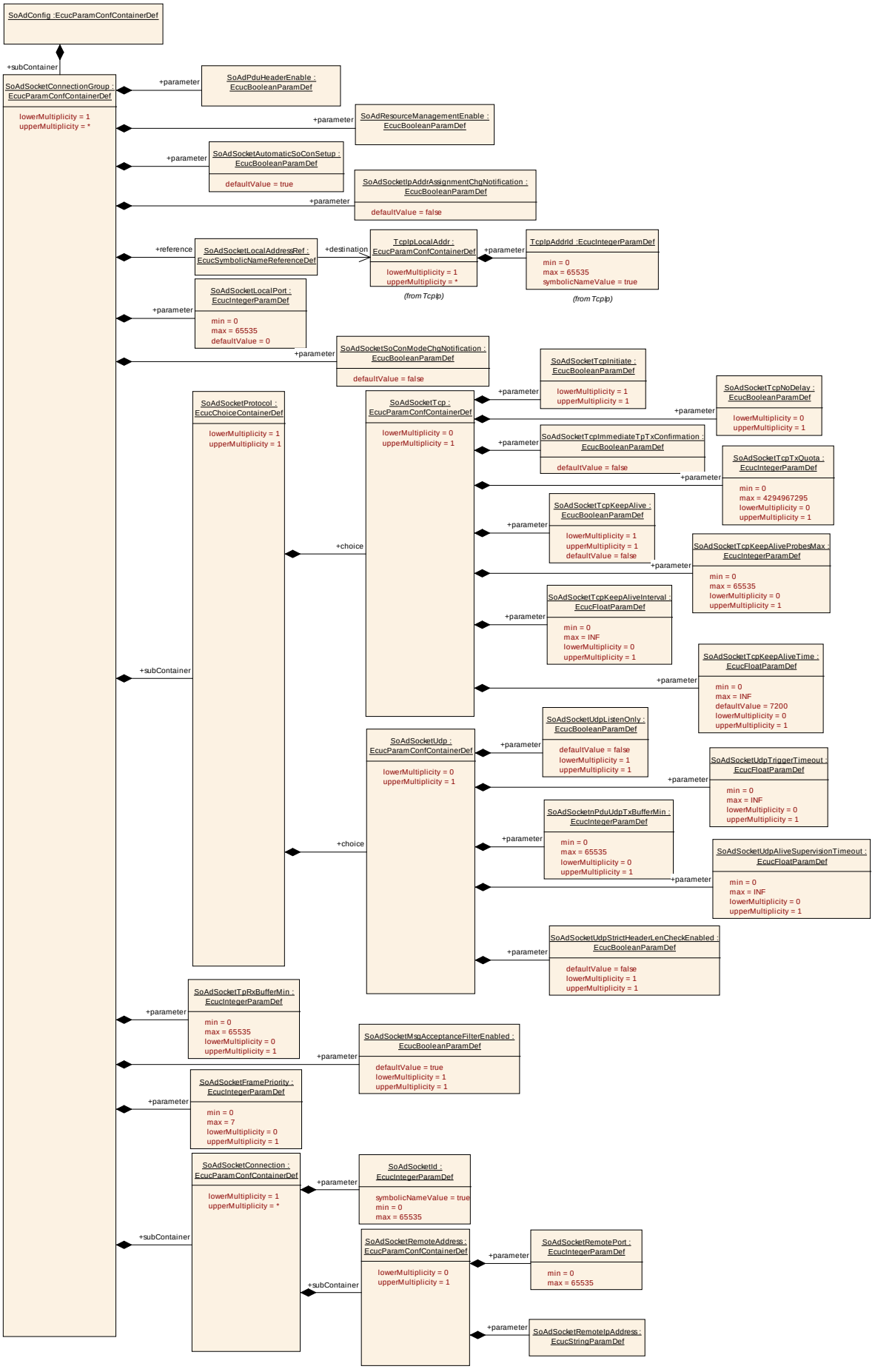

SoAdConfig :EcucParamConfContainerDef
+subContainer
SoAdSocketConnectionGroup : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
SoAdPduHeaderEnable : EcucBooleanParamDef
+parameter
SoAdResourceManagementEnable : EcucBooleanParamDef
+parameter
SoAdSocketAutomaticSoConSetup : EcucBooleanParamDef
defaultValue = true
+parameter
SoAdSocketIpAddrAssignmentChgNotification : EcucBooleanParamDef
defaultValue = false
+reference
SoAdSocketLocalAddressRef : EcucSymbolicNameReferenceDef
+destination
TcpIpLocalAddr : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from TcpIp)
+parameter
TcpIpAddrId :EcucIntegerParamDef
min = 0
max = 65535
symbolicNameValue = true
(from TcpIp)
+parameter
SoAdSocketLocalPort : EcucIntegerParamDef
min = 0
max = 65535
defaultValue = 0
+parameter
SoAdSocketSoConModeChgNotification : EcucBooleanParamDef
defaultValue = false
SoAdSocketProtocol : EcucChoiceContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
SoAdSocketTcp : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketTcpInitiate : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
SoAdSocketTcpNoDelay : EcucBooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketTcpImmediateTpTxConfirmation : EcucBooleanParamDef
defaultValue = false
+parameter
SoAdSocketTcpTxQuota : EcucIntegerParamDef
min = 0
max = 4294967295
lowerMultiplicity = 0
upperMultiplicity = 1
+choice
+parameter
SoAdSocketTcpKeepAlive : EcucBooleanParamDef
lowerMultiplicity = 1
upperMultiplicity = 1
defaultValue = false
+parameter
SoAdSocketTcpKeepAliveProbesMax : EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketTcpKeepAliveInterval : EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketTcpKeepAliveTime : EcucFloatParamDef
min = 0
max = INF
defaultValue = 7200
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
SoAdSocketUdp : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketUdpListenOnly : EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
SoAdSocketUdpTriggerTimeout : EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+choice
+parameter
SoAdSocketnPduUdpTxBufferMin : EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketUdpAliveSupervisionTimeout : EcucFloatParamDef
min = 0
max = INF
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketUdpStrictHeaderLenCheckEnabled : EcucBooleanParamDef
defaultValue = false
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
SoAdSocketTpRxBufferMin : EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketMsgAcceptanceFilterEnabled : EcucBooleanParamDef
defaultValue = true
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
SoAdSocketFramePriority : EcucIntegerParamDef
min = 0
max = 7
lowerMultiplicity = 0
upperMultiplicity = 1
SoAdSocketConnection : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
SoAdSocketId : EcucIntegerParamDef
symbolicNameValue = true
min = 0
max = 65535
+subContainer
SoAdSocketRemoteAddress : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
SoAdSocketRemotePort : EcucIntegerParamDef
min = 0
max = 65535
+subContainer
+parameter
SoAdSocketRemoteIpAddress : EcucStringParamDef

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

SWS Item	**ECUC_SoAd_00020 :**		
Name	SoAdSocketRemotePort		
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	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: 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		
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	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: Parameter shall be configured for SoAdSocketRoute that is related to a SoAdSocketConnection belonging to a SoAdSocketConnectionGroup with SoAdPduHeaderEnable set to TRUE.		

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

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

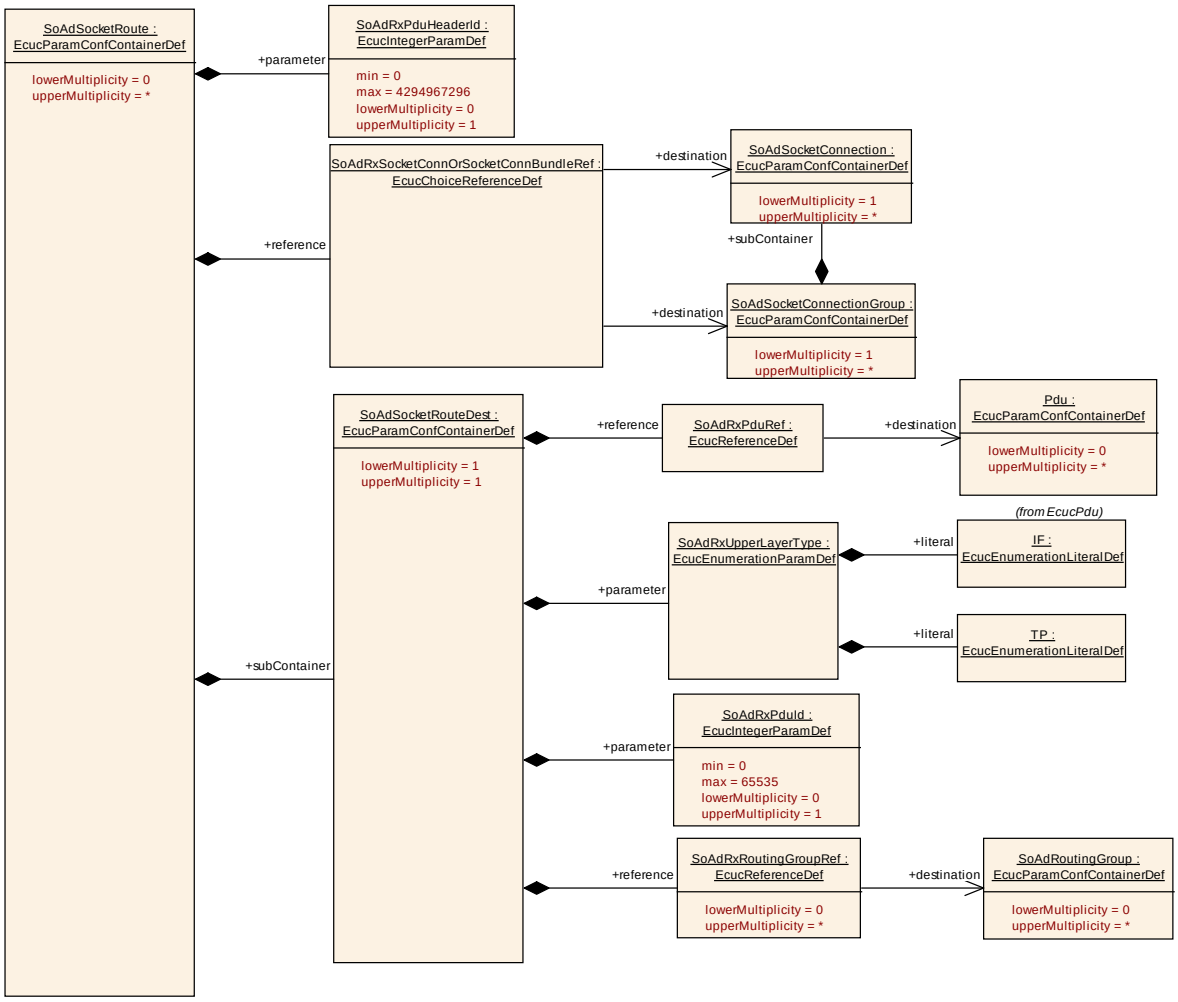


Figure 6: Socket Route configuration

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 TcpIp socket.		
Configuration Parameters			

SWS Item	**ECUC_SoAd_00116 :**		
Name	SoAdRxPduId		
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	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

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

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

Scope / Dependency	scope: ECU

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

No Included Containers

10.2.14 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		
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	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_SoAd_00118 :**	
Name	SoAdTxUpperLayerType	
Description	Specifies the upper layer interface type (must be "IF" in case of multiple PduRoutes).	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	IF	If Interface
	TP	TP Interface

Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

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

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

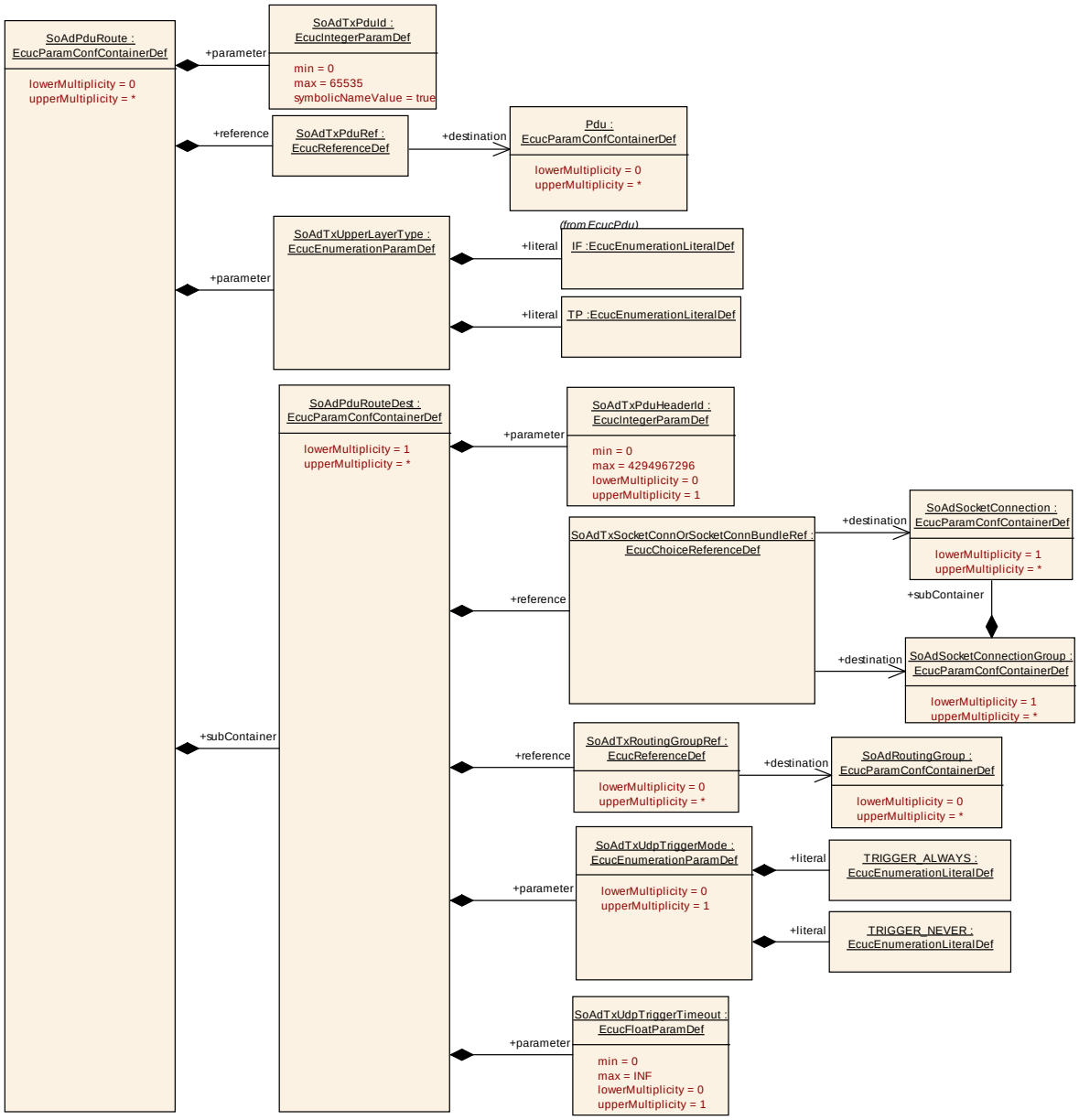


Figure 7: Pdu Route configuration

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

SWS Item	**ECUC_SoAd_00136 :**		
Name	SoAdTxUdpTriggerMode		
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
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: SoAdSocketProtocol, SoAdTxUpperLayerType

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

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

SWS Item	**ECUC_SoAd_00034 :**		
Name	SoAdTxSocketConnOrSocketConnBundleRef		
Description	Choice Reference to a SocketConnection or to a SocketConnectionGroup on which the PDU is to be sent on. The reference to a SocketConnectionGroup shall only be used for upper layers with IF API.		
Multiplicity	1		
Type	Choice reference to [SoAdSocketConnection , SoAdSocketConnectionGroup]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local dependency: SoAdTxUpperLayerType

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

SWS Item	**ECUC_SoAd_00122 :**		
Name	SoAdRoutingGroupIsEnabledAtInit		
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	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_SoAd_00146 :**		
Name	SoAdRoutingGroupTxTriggerable		
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		
Post-Build Variant Value	false		
Value Configuration Class	***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, SRS_BSW_00336, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00160, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00307, SRS_BSW_00335, SRS_BSW_00410, SRS_BSW_00314, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00006, SRS_BSW_00306, SRS_BSW_00309, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334)