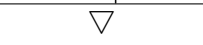
Document Title	Specification of Socket Adaptor
Document Category	Software specification (SWS)
Document UID	416
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Added support for secure UDP communication using DTLS • Added support for DMA-based data transfer • Removed the configuration switch for the version information API • Restructured the API specification and removed obsolete chapters
2025-11-27	R25-11	AUTOSAR Release Management	• Enhanced TCP retransmission handling. • Removed Path MTU discovery.
2024-11-27	R24-11	AUTOSAR Release Management	• Replaced `Sd_AclPolicyCheck()` by `Sd_RequestRoutingGroup-Enable()`. • Added `SoAdSocketTimeToLive`. • Corrected occurrence of TpTp in `<Up>_[SoAd][Tp]TxConfirmation()` and `<Up>_[SoAd][Tp]RxIndication()`
2023-11-23	R23-11	AUTOSAR Release Management	• Added `SoAdSocketIpAddrAssignmentChg-NotifUpperLayerRef`. • Added ACL support. • Removed module prefix from security events. • Improved TCP stream handling. • Improved nPdu transmit requirements [SWS_SoAd_00734] and [SWS_SoAd_00747]. • Resolved contradiction in [SWS_SoAd_00683].



△

2022-11-24	R22-11	AUTOSAR Release Management	• Improved SoAd_TcpIpEvent() requirement [SWS_SoAd_00688]. • Improved nPdu requirements [SWS_SoAd_00691] and [SWS_SoAd_00735]. • Added NA infix to "not applicable" requirement [SWS_SoAd_NA_00296].
2021-11-25	R21-11	AUTOSAR Release Management	• Introduced config parameter SoAdSocketSoConModeChgBswMNotification • Added limitations for SomeIP protocol handling. • Introduced config parameter SoAdSocketTcpRetransmissionTimeout • Introduced config parameter SoAdSocketTcpAutoConnectTimeout • Introduced SoAd_IsConnectionReady() to retrieve connection status from TcpIp
2020-11-30	R20-11	AUTOSAR Release Management	• Introduction of Security Event reporting • Added limitations for checking SomeIP protocol header. • Added state switch in context of SoAd_ReleaseRemoteAddr()
2019-11-28	R19-11	AUTOSAR Release Management	• Support for selectable PDU reception paths for multiple instances of the same Sd service • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Introduction of Transport Layer Security - TLS (DRAFT) • minor corrections / clarifications / editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Rollout of Runtime Errors • Clarifications and corrections of requirements • Editorial changes

▽

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Support for decoupled data transmission • Optimization for Client/Server communication • Introduction of reliable TxConfirmations • Clarifications and corrections of requirements
2015-07-31	4.2.2	AUTOSAR Release Management	• editorial Clarifications and corrections of requirements • Editorial changes
2014-10-31	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
2014-03-31	4.1.3	AUTOSAR Release Management	• TP API: Harmonization of ChangeParameter function • Clarifications and corrections of requirements and sequence charts • Editorial changes
2013-10-31	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
2013-03-15	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

▽

△

2011-12-22	4.0.3	AUTOSAR Administration	• Rectify inconsistencies in synchronicity and reentrancy • Adjust parameter multiplicity • New traceability mechanism
2010-09-30	3.1.5	AUTOSAR Administration	• ComStack Harmonization. • Allow for Post-Build Configuration • API for IP address change notification • Allow full handling of TCP connections
2010-02-02	3.1.4	AUTOSAR Administration	• Initial Release Revision

Disclaimer

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

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

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

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 11
2 Acronyms and Abbreviations 13
3 Related documentation 14
3.1 Input documents & related standards and norms . . . 14
3.2 Related specification . . . 14
4 Constraints and assumptions 15
4.1 Limitations . . . 15
4.2 Applicability to car domains . . . 16
5 Dependencies to other modules 17
5.1 AUTOSAR TCP/IP Stack . . . 17
5.2 Generic Upper Layer . . . 17
5.3 File structure . . . 17
5.3.1 Code file structure . . . 17
6 Requirements Tracing 18
7 Functional specification 19
7.1 Socket Connections . . . 19
7.1.1 Socket Connection Open . . . 20
7.1.2 Socket Connection Close . . . 26
7.1.3 Socket Connection Open/Close Sequence Remarks . . . 29
7.1.4 Notifications . . . 30
7.1.5 Connection Status . . . 30
7.2 PDU Transmission . . . 31
7.2.1 PDU Transmission via IF-API . . . 31
7.2.1.1 Local buffer copy using asynchronous copy . . . 33
7.2.2 PDU Transmission via IF-API and nPduUdpTxBuffer . . . 34
7.2.3 DTLS over UDP . . . 37
7.2.4 PDU Transmission via IfRoutingGroupTransmit API . . . 38
7.2.5 PDU Transmission via TP-API . . . 38
7.3 PDU Header option . . . 41
7.3.1 Scenario coverage: large data transmission via IF API over TCP . . . 41
7.4 PDU Reception . . . 42
7.4.1 PDU Reception via IF-API . . . 46
7.4.2 PDU Reception via TP-API (PDU Header disabled) . . . 46
7.4.3 PDU Reception via TP-API (PDU Header enabled) . . . 48
7.5 Best Match Algorithm . . . 49
7.6 Message Acceptance Policy . . . 50
7.7 TP PDU Cancelation . . . 51
7.8 Disconnection and recovery . . . 51

7.9 Routing Groups . . . 52
7.10 PDU fan-out . . . 53
7.11 Buffer handling . . . 54
7.12 Security Events . . . 54
7.13 Error Classification . . . 55
7.13.1 Development Errors . . . 55
7.13.2 Runtime Errors . . . 56
7.13.3 Production Errors . . . 56
8 API specification 57
8.1 Imported types . . . 57
8.2 Type definitions . . . 57
8.2.1 SoAd_SoConIdType . . . 57
8.2.2 SoAd_SoConModeType . . . 58
8.2.3 SoAd_RoutingGroupIdType . . . 58
8.2.4 SoAd_ConfigType . . . 59
8.2.5 SoAd_MeasurementIdxType . . . 59
8.3 Function definitions . . . 59
8.3.1 General . . . 60
8.3.1.1 SoAd_Init . . . 60
8.3.2 Normal Operation . . . 60
8.3.2.1 SoAd_IfTransmit . . . 60
8.3.2.2 SoAd_IfRoutingGroupTransmit . . . 61
8.3.2.3 SoAd_IfSpecificRoutingGroupTransmit . . . 62
8.3.2.4 SoAd_TpTransmit . . . 63
8.3.2.5 SoAd_ReleaseRxBuffer . . . 64
8.3.3 Transmit/Receive Cancelation API . . . 64
8.3.3.1 SoAd_TpCancelTransmit . . . 64
8.3.3.2 SoAd_TpCancelReceive . . . 65
8.3.4 Information and Control API . . . 65
8.3.4.1 SoAd_GetSoConId . . . 65
8.3.4.2 SoAd_OpenSoCon . . . 66
8.3.4.3 SoAd_CloseSoCon . . . 67
8.3.4.4 SoAd_GetSoConMode . . . 68
8.3.4.5 SoAd_RequestIpAddrAssignment . . . 68
8.3.4.6 SoAd_ReleaseIpAddrAssignment . . . 69
8.3.4.7 SoAd_GetLocalAddr . . . 70
8.3.4.8 SoAd_GetPhysAddr . . . 71
8.3.4.9 SoAd_GetRemoteAddr . . . 72
8.3.4.10 SoAd_EnableRouting . . . 73
8.3.4.11 SoAd_EnableSpecificRouting . . . 73
8.3.4.12 SoAd_DisableRouting . . . 74
8.3.4.13 SoAd_DisableSpecificRouting . . . 75
8.3.4.14 SoAd_SetRemoteAddr . . . 75

8.3.4.15 SoAd_SetUniqueRemoteAddr . . . 77
8.3.4.16 SoAd_ReleaseRemoteAddr . . . 78
8.3.4.17 SoAd_TpChangeParameter . . . 79
8.3.4.18 SoAd_ReadDhcpHostNameOption . . . 80
8.3.4.19 SoAd_WriteDhcpHostNameOption . . . 81
8.3.4.20 SoAd_GetAndResetMeasurementData . . . 82
8.3.4.21 SoAd_IsConnectionReady . . . 84
8.3.5 TCP/IP Stack . . . 84
8.3.5.1 SoAd_RxIndication . . . 84
8.3.5.2 SoAd_CopyTxData . . . 85
8.3.5.3 SoAd_TxConfirmation . . . 86
8.3.5.4 SoAd_TcpAccepted . . . 87
8.3.5.5 SoAd_TcpConnected . . . 88
8.3.5.6 SoAd_TcpIpEvent . . . 88
8.3.5.7 SoAd_LocalIpAddrAssignmentChg . . . 90
8.4 Scheduled functions . . . 90
8.4.1 Terms and definitions . . . 90
8.4.2 SoAd_MainFunction . . . 91
8.5 Expected interfaces . . . 91
8.5.1 Mandatory interfaces . . . 91
8.5.2 Optional interfaces . . . 92
8.5.3 Configurable interfaces . . . 93
8.5.3.1 <Up>_[SoAd][If]RxIndication . . . 94
8.5.3.2 <Up>_[SoAd][If]TriggerTransmit . . . 94
8.5.3.3 <Up>_[SoAd][If]TxConfirmation . . . 95
8.5.3.4 <Up>_[SoAd][Tp]StartOfReception . . . 95
8.5.3.5 <Up>_[SoAd][Tp]CopyRxData . . . 96
8.5.3.6 <Up>_[SoAd][Tp]RxIndication . . . 97
8.5.3.7 <Up>_[SoAd][Tp]CopyTxData . . . 97
8.5.3.8 <Up>_[SoAd][Tp]TxConfirmation . . . 98
8.5.3.9 <Up>_SoConModeChg . . . 99
8.5.3.10 <Up>_LocalIpAddrAssignmentChg . . . 99
9 Sequence diagrams and Transition Tables 101
9.1 Address - Assignment . . . 101
9.2 Socket Connection Setup - UDP . . . 102
9.3 Socket Connection Setup - TCP . . . 103
9.4 Reception - Upper Layer If API . . . 105
9.5 Reception - Upper Layer TP API . . . 106
9.6 Transmission - Upper Layer If API - TCP . . . 108
9.7 Transmission - Upper Layer If API - UDP . . . 109
9.8 Transmission - Upper Layer TP API . . . 110

10 Configuration specification 111
10.1 How to read this chapter . . . 111
10.2 Containers and configuration parameters . . . 111
10.2.1 SoAd . . . 111
10.2.2 SoAdBswModules . . . 112
10.2.3 SoAdGeneral . . . 116
10.2.4 SoAdSecurityEventRefs . . . 120
10.2.5 SoAdConfig . . . 123
10.2.6 SoAdSocketConnectionGroup . . . 124
10.2.7 SoAdSocketConnection . . . 132
10.2.8 SoAdSocketProtocol . . . 133
10.2.9 SoAdSocketUdp . . . 133
10.2.10 SoAdSocketTcp . . . 137
10.2.11 SoAdSocketRemoteAddress . . . 144
10.2.12 SoAdSocketRoute . . . 146
10.2.13 SoAdSocketRouteDest . . . 148
10.2.14 SoAdPduRoute . . . 149
10.2.15 SoAdPduRouteDest . . . 153
10.2.16 SoAdRoutingGroup . . . 155
10.3 Configuration constraints . . . 157
10.4 Published Information . . . 158
A Not applicable requirements 159
B Change history of AUTOSAR traceable items 160
B.1 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 160
B.1.1 Added Specification Items in R23-11 . . . 160
B.1.2 Changed Specification Items in R23-11 . . . 160
B.1.3 Deleted Specification Items in R23-11 . . . 160
B.2 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 160
B.2.1 Added Specification Items in R24-11 . . . 160
B.2.2 Changed Specification Items in R24-11 . . . 160
B.2.3 Deleted Specification Items in R24-11 . . . 161
B.3 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 161
B.3.1 Added Specification Items in R25-11 . . . 161
B.3.2 Changed Specification Items in R25-11 . . . 161
B.3.3 Deleted Specification Items in R25-11 . . . 161
B.4 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 161
B.4.1 Added Specification Items in R26-11 . . . 161
B.4.2 Changed Specification Items in R26-11 . . . 161

B.4.3 Deleted Specification Items in R26-11 162
B.4.4 Added Constraints in R26-11 162
B.4.5 Changed Constraints in R26-11 162
B.4.6 Deleted Constraints in R26-11 162

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 predetermined 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.

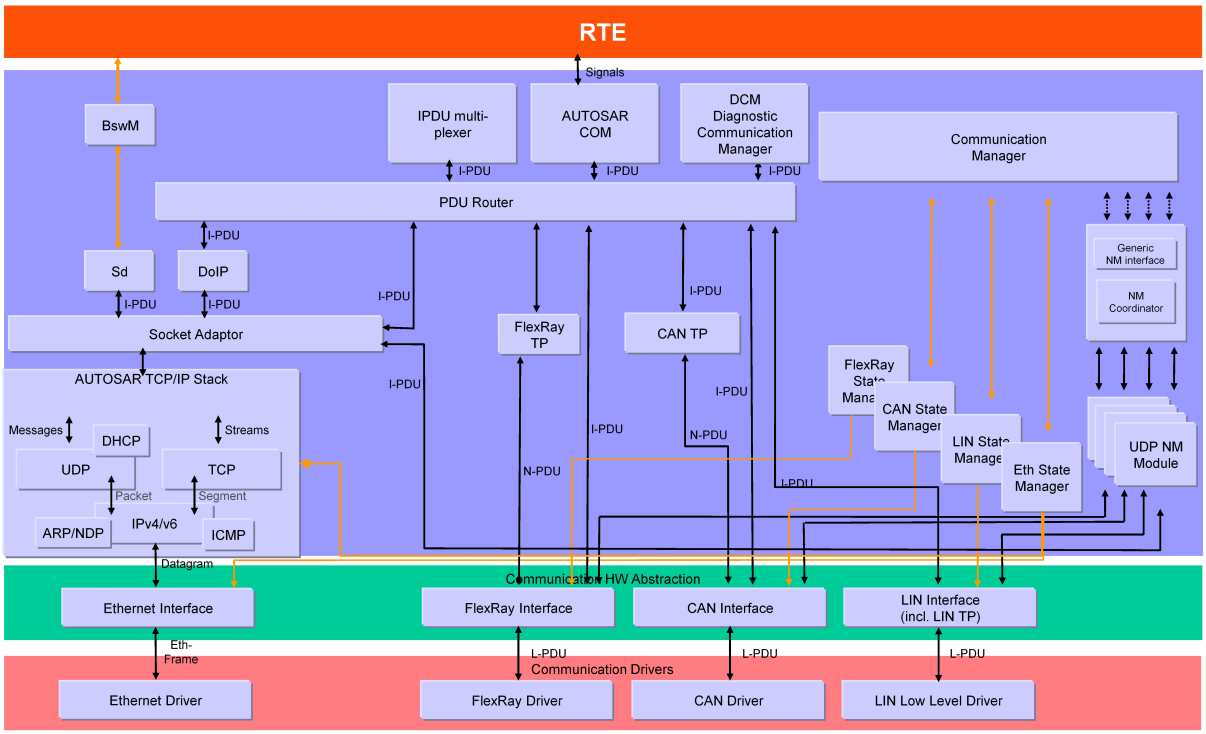


Figure 1.1: Extended AUTOSAR Communication Stack.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the SoAd module that are not included in the [1, AUTOSAR glossary].

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
TLS	Transport Layer Security
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 & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[4] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter

[5] Specification of UDP Network Management
AUTOSAR_CP_SWS_UDPNetworkManagement

[6] Specification of Module XCP
AUTOSAR_CP_SWS_XCP

[7] Specification of Service Discovery
AUTOSAR_CP_SWS_ServiceDiscovery

[8] Specification of Diagnostic over IP
AUTOSAR_CP_SWS_DiagnosticOverIP

[9] The Dynamic Host Configuration Protocol (DHCP) Client Fully Qualified Domain Name (FQDN) Option
https://rfc-editor.org/rfc/rfc4702.txt

[10] The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option
https://rfc-editor.org/rfc/rfc4704.txt

[11] Specification of ECU Configuration
AUTOSAR_CP_TPS_ECUConfiguration

[12] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for Socket Adaptor.

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

4 Constraints and assumptions

4.1 Limitations

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

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

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

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

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

SOME/IP Protocol Specification [3] specifies to check Protocol Version prior to the check of Service ID and Method ID to be valid. SoAd does not check Protocol Version. Independent of future version It will always interpret the first 4 bytes as message ID (header ID in this document) and the second 4 bytes as payload length.

SOME/IP Protocol Specification [3] specifies to check Interface Version prior to the check of Method ID to be valid. SoAd does not check Interface Version. SoAd performs routing to configured Message IDs independent of the contained Interface Version on dedicated routing paths.

SOME/IP Protocol Specification [3] specifies to check Service ID and Method ID to be valid. Invalid IDs shall be responded with corresponding error codes.

SoAd checks the SOME/IP message ID (header ID in this document) which is a combination of service ID and method ID. If the ID is not valid the module is not capable to respond with corresponding error codes. Instead, runtime error `SOAD_E_INV_PDUHEADER_ID` is raised.

SOME/IP Protocol Specification [3] specifies with [PRS_SOMEIP_00535] that all Transport Protocol Bindings shall support transporting more than one SOME/IP message in a Transport Layer PDU. In case of TCP this requirement is out of SoAd scope since the message packing and transportation over a TCP stream can not be influenced by SoAd. In case of UDP collecting of SOME/IP message is limited to IF API only. According to [SWS_SoAd_00553] a SOME/IP message will be retrieved from

upper layer via TP API and immediately sent over UDP. Packing of messages is not foreseen and not supported in this case.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to 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, TLS, 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) [4]: IF-PDU and TP-PDU API
- UDP Network Management (UdpNm) [5]: IF-PDU API
- XCP on Ethernet (Xcp) [6] : IF-PDU API
- Service Discovery (Sd) [7]: IF-PDU API, Control API 8.3.4
- Diagnostics over IP (DoIP) [8]: 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 [2] Chapter 5.1.6 *“Code file structure”*.

6 Requirements Tracing

Requirement	Description	Satisfied by
[SRS_BSW_00337]	Classification of development errors	[SWS_SoAd_00732] [SWS_SoAd_00744]
[SRS_Eth_00017]	TCP shall be implemented according to IETF RFC 793	[SWS_SoAd_00765] [SWS_SoAd_00766]
[SRS_Eth_00058]	SoAd shall support generic upper layers	[SWS_SoAd_00741]
[SRS_Eth_00085]	Robustness aganist the change of logical addresses	[SWS_SoAd_00532] [SWS_SoAd_00694] [SWS_SoAd_00695] [SWS_SoAd_00742] [SWS_SoAd_00743] [SWS_SoAd_00745] [SWS_SoAd_00746] [SWS_SoAd_00762]
[SRS_Eth_00116]	The Socket Adaptor shall implement a mechanism to transmit multiple PDUs within the same UDP datagram	[SWS_SoAd_00546] [SWS_SoAd_00547] [SWS_SoAd_00548] [SWS_SoAd_00549] [SWS_SoAd_00550] [SWS_SoAd_00683] [SWS_SoAd_00684] [SWS_SoAd_00685] [SWS_SoAd_00690] [SWS_SoAd_00691] [SWS_SoAd_00696] [SWS_SoAd_00697] [SWS_SoAd_00734] [SWS_SoAd_00735] [SWS_SoAd_00736] [SWS_SoAd_00747]
[SRS_Eth_00124]	The SoAd shall implement mechanisms to share the same PDU pair for the reception from and transmission to multiple remote nodes	[SWS_SoAd_00738] [SWS_SoAd_00739] [SWS_SoAd_00740]
[SRS_Eth_00131]	The SoAd shall support access to measurement counter values	[SWS_SoAd_00748] [SWS_SoAd_00749] [SWS_SoAd_00750] [SWS_SoAd_00751] [SWS_SoAd_00752] [SWS_SoAd_00753] [SWS_SoAd_00754] [SWS_SoAd_00755] [SWS_SoAd_00756] [SWS_SoAd_00757] [SWS_SoAd_00758]
[SRS_Eth_00163]	Method Call Request Check	[SWS_SoAd_00772] [SWS_SoAd_00773] [SWS_SoAd_00774] [SWS_SoAd_00775]

Table 6.1: Requirements Tracing

7 Functional specification

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

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

7.1 Socket Connections

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

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

connections with identical connection parameters can be grouped to socket connection groups.

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

[SWS_SoAd_00743]

Upstream requirements: SRS_Eth_00085

⌈SoAd shall lock the remote address during the following situations:

1. TCP socket connections not in state SOAD_SOCON_OFFLINE,
2. active receptions,
3. pending receptions of TP-PDUs,
4. active transmissions,
5. pending transmissions of the nPdu feature,
6. pending transmissions initiated via SoAd_IfRoutingGroupTransmit() or SoAd_IfSpecificRoutingGroupTransmit().

A locked remote address can't be modified by upper layers.⌋

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_00765]

Upstream requirements: SRS_Eth_00017

⌈If SoAdSocketTcpAutoConnectTimeout has been configured for that socket connection to a value > 0 SoAd shall repeat the invocation stated in [SWS_SoAd_00590] bullet (4) only for the time specified in SoAdSocketTcpAutoConnectTimeout.⌋

[SWS_SoAd_00766]

Upstream requirements: SRS_Eth_00017

⌈If SoAdSocketTcpAutoConnectTimeout has elapsed for that socket connection and the socket connection has not yet reached SOAD_SOCON_ONLINE, then

1. the runtime error SOAD_E_TCP_AUTOCONNECT_FAILED shall be raised.
2. the socket connection shall close related TcpIp socket.
3. the socket connection shall be put into SOAD_SOCON_OFFLINE.
4. the socket connection shall stay in SOAD_SOCON_OFFLINE.

⌋

Note: Socket connections with elapsed SoAdSocketTcpAutoConnectTimeout will become inactive until module is reset.

[SWS_SoAd_00638] ⌈SoAd shall perform the following actions within SoAd_MainFunction() to open a TCP socket connection with SoAdSocketTcpInitiate set to FALSE:

1. In case no Listen-Socket is assigned to the socket connection:
   (a) Get an appropriate socket from 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 `SoAdSocketTcpInitiate` 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.
8. In case of a TCP socket: `TCPIP_PARAMID_TCP_OPTIONFILTER` and the value of `TcpIpTcpConfigOptionFilterId` specified in `TcpIpTcpConfigOptionFilter` referenced by `SoAdSocketTCPOptionFilterRef` if the optional parameter is enabled.
9. `TCPIP_PARAMID_FLOWLABEL` and the value specified by `SoAdSocketFlowLabel` if the optional parameter is enabled.
10. `TCPIP_PARAMID_DSCP` and the value specified by `SoAdSocketDifferentiatedServicesField` if the optional parameter is enabled.
11. In case of a UDP socket: `TCPIP_PARAMID_UDP_CHECKSUM` and the value of `SoAdSocketUdpChecksumEnabled`.
12. In case of a TCP socket: If `SoAdSocketTcpTlsConnectionRef` is defined the function shall be called with the parameter ID `TCPIP_PARAMID_TLS_CONNECTION_ASSIGNMENT` and the value from this reference as the parameter value to assign a TLS connection to this socket.

13. In case of a UDP socket: If SoAdDatagramTlsConnectionRef is defined the function shall be called with the parameter ID TCPIP_PARAMID_TLS_CONNECTION_ASSIGNMENT and the value from this reference as the parameter value to assign a DTLS-capable TLS connection to this socket.
14. In case of a TCP socket: TCPIP_PARAMID_TCP_RETRANSMIT_TIMEOUT and the value specified by SoAdSocketTcpRetransmissionTimeout if the optional parameter is enabled. If SoAdSocketTcpRetransmissionTimeout is configured to INF, SoAd shall use the corresponding value representing INF from TcpIp module.
15. In case of a TCP socket: TCPIP_PARAMID_TCP_RMAXRTX and the value specified by SoAdSocketTcpMaxRetransmissions if the optional parameter is enabled.
16. In case of a TCP socket: TCPIP_PARAMID_IMMEDIATE_TX_CONFIRMATION and the value derived from SoAdSocketIfImmediateTxConfirmation. If SoAdSocketIfImmediateTxConfirmation is TRUE, SoAd shall configure TcpIp for immediate IF transmit confirmation behavior.
17. In case of a TCP socket: TCPIP_PARAMID_TCP_MAX_RETRANSMIT_TIMEOUT and the value specified by SoAdSocketTcpMaxRetransmissionTimeout if the optional parameter is enabled. If SoAdSocketTcpRetransmissionTimeout is configured to INF, SoAd shall use the corresponding value representing INF from TcpIp module.
18. TCPIP_PARAMID_TTL and the value specified by SoAdSocketTimeToLive 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 with SoAdDatagramTlsConnectionRef not set 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_Close() 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_Close() 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_Close() 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_Close() 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_00802] Close DTLS session on remote address switch ⌈If a call of SoAd_ReleaseRemoteAddr() or SoAd_SetRemoteAddr(), or a SoAd-internal remote-address reset path (for example as specified by [SWS_SoAd_00695]), changes the remote address or port of a socket connection with an established DTLS connection, SoAd shall call TcpIp_DtlsClose().⌋

Note: DTLS closure for socket-connection close scenarios is covered via [SWS_SoAd_00642] calling TcpIp_Close(), with DTLS-session termination behavior specified by the TcpIp DTLS close semantics. DTLS closure due to lower-layer DTLS issues is handled via TCPIP_UDP_DTLS_STATE_CLOSED callback processing in the DTLS close-state callback path.

[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] ⌈If SoAd_TcpIpEvent() is called with Event set to TCPIP_TCP_FIN_RECEIVED, SoAd shall perform the closure according to [SWS_SoAd_00637]. 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_00741]

Upstream requirements: SRS_Eth_00058

⌈Each time a socket connection state changes, SoAd shall notify the upper layer specified by `SoAdSocketSoConModeChgNotifUpperLayerRef` with the configured upper layer notification function `<Up>_SoConModeChg()` and the new state if the optional reference is set for the socket connection.⌋

[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.⌋

[SWS_SoAd_00767] ⌈Each time a socket connection state changes, SoAd shall notify the BswM of the socket connection state change with the notification function `BswM_SoAd_SoConModeChg()` and the new state if `SoAdSocketSoConModeChgBswMNotification` is set to `TRUE` for the socket connection.⌋

[SWS_SoAd_00776]

Status: DRAFT

⌈Each time the IP address assignment related to socket connection changes, SoAd shall notify the upper layer specified by `SoAdSocketIpAddrAssignmentChgNotifUpperLayerRef` with the configured upper layer notification function `<Up>_LocalIpAddrAssignmentChg()` and the new address state if the optional reference is set for the socket connection.⌋

7.1.5 Connection Status

[SWS_SoAd_00768] ⌈If `SoAd_IsConnectionReady()` is called and the referred socket connection has no socket assigned, SoAd shall return `TCPIP_E_NOT_OK`.⌋

[SWS_SoAd_00769] ⌈If `SoAd_IsConnectionReady()` is called and the referred socket connection has a socket assigned, SoAd shall call `TcpIp_IsConnectionReady()` to pass `RemoteAddrPtr` for the corresponding socket. The return value of `TcpIp_IsConnectionReady()` shall be forwarded to the caller.⌋

7.2 PDU Transmission

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

The upper layer module of the SoAd may use the Interface (IF) API or the Transport Protocol (TP) API for the transmit request and data provision respectively.

7.2.1 PDU Transmission via IF-API

[SWS_SoAd_00539] ⌈For the transmission of a PDU requested by an upper layer using the IF-API, the SoAd shall

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

⌋

[SWS_SoAd_00738]

Upstream requirements: SRS_Eth_00124

⌈If development error detection is enabled: In case of a transmit request for a SoAdPduRoute that refers to a global PDU structure configured with a MetaDataItem of the type SOCKET_CONNECTION_ID_16 and the contained SoAdPduRouteDest refers to a socket connection group, SoAd shall raise the development error SOAD_E_INV_METADATA, if the socket connection identified by PduInfoType.MetaDataPtr is not part of the socket connection group of the related SoAdPduRouteDest.⌋

[SWS_SoAd_00739]

Upstream requirements: SRS_Eth_00124

⌈In case of a transmit request for a SoAdPduRoute that refers to a global PDU structure configured with a MetaDataItem of the type SOCKET_CONNECTION_ID_16 and the contained SoAdPduRouteDest refers to a socket connection group, SoAd shall only perform the transmission on the socket connection identified by PduInfoType.MetaDataPtr instead of the whole group.⌋

[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: TxPduId 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.⌋

Note: SoAd_CopyTxData() is a callback from TcpIp to SoAd and shall not be triggered by SoAd itself. SoAd triggers transmission via TcpIp_UdpTransmit()() or TcpIp_TcpTransmit()(); TcpIp decides whether and when SoAd_CopyTxData() is invoked.

[SWS_SoAd_00731] ⌈If SoAd_IfTransmit() was called with PduInfoPtr->SduDataPtr set to NULL_PTR, SoAd shall use <Up>_[SoAd][If]TriggerTransmit() to retrieve the PDU data from the upper layer.⌋

[SWS_SoAd_00544] ⌈In case of a UDP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (<Up>_[SoAd][If]TxConfirmation()) with result set to E_OK within the next SoAd_MainFunction() after the latest TcpIp_UdpTransmit() call returning successfully.⌋

[SWS_SoAd_00545] ⌈In case of a TCP socket connection the SoAd shall call the upper layer with the configured transmit confirmation function (<Up>_[SoAd][If]TxConfirmation()) with result set to E_OK within the SoAd_TxConfirmation() callback function after all PDU data (from one or multiple transmit requests) have been confirmed for transmission.⌋

Note: there is only a single confirmation even in case of multiple transmit requests for the same PDU, i.e. in case a further transmit is requested for the same PDU on a TCP socket connection before the last request is completed, there is no separate confirmation for the last request, but only a final confirmation for all PDU data.

7.2.1.1 Local buffer copy using asynchronous copy

[SWS_SoAd_00778] Local buffer copy with asynchronous copy above threshold ⌈If `KeepLocalPduBuffer` is set to `TRUE` for a received IF PDU and `SduLength` is greater than `EcucDmaThreshold`, SoAd shall process the copy path asynchronously.⌋

Note: Please refer to [2, BSW General] Chapter 7.1.21 *"Asynchronous copy approach (DRAFT)"* for more details on asynchronous copying.

[SWS_SoAd_00779] Local buffer copy at or below asynchronous-copy threshold ⌈If `KeepLocalPduBuffer` is set to `TRUE` for a received IF PDU and `SduLength` is less than or equal to `EcucDmaThreshold`, SoAd shall process the copy path synchronously (CPU copy).⌋

[SWS_SoAd_00780] KeepLocalPduBuffer disabled behavior ⌈If `KeepLocalPduBuffer` is set to `FALSE` for an IF PDU, SoAd shall use synchronous copy processing in caller context for transmission requests and reception indications, and shall not require retention of local transmission-buffer resources for the corresponding flow.⌋

Note: The TP data exchange APIs `<Up>_[SoAd][Tp]CopyTxData()` and `<Up>_[SoAd][Tp]CopyRxData()` are synchronous copy APIs executed in caller context. Therefore, applying asynchronous-copy handling to these TP paths would not provide a useful benefit.

[SWS_SoAd_00781] Transmission buffer retention with KeepLocalPduBuffer ⌈For transmission with `KeepLocalPduBuffer` set to `TRUE`, SoAd shall keep its retained local transmission-buffer references and related internal transmission resources valid until `SoAd_TxConfirmation` is received for the associated transmission instance.⌋

[SWS_SoAd_00782] Resource release before upper-layer confirmation ⌈On `SoAd_TxConfirmation`(`SocketId`, `Length`), SoAd shall release all retained internal transmission resources associated with the confirmed transmission instance on the given `SocketId` and shall handle IF confirmation as follows:

1. if `Length` is greater than 0 and `SoAdSocketIfImmediateTxConfirmation` is `FALSE`, SoAd shall call `<Up>_[SoAd][If]TxConfirmation()` for the related PDU;
2. if `Length` is greater than 0 and `SoAdSocketIfImmediateTxConfirmation` is `TRUE`, SoAd shall not issue an additional IF transmit confirmation for the same transmission instance in this callback context;
3. if `Length` is 0, SoAd shall report runtime error `SOAD_E_MESSAGE_LOST`. If `SoAdSocketIfImmediateTxConfirmation` is `FALSE`, SoAd shall call `<Up>_[SoAd][If]TxConfirmation()` with result `E_NOT_OK`; if `SoAdSocketIfImmediateTxConfirmation` is `TRUE`, SoAd shall not issue an additional IF transmit confirmation in this callback context.

⌋

Note: SoAd_TxConfirmation is the lower-layer callback from TcpIp to SoAd. The SoAd-to-upper-layer API for IF transmit confirmation is <Up>_[SoAd][If]TxConfirmation(). Upper-layer specific buffer lifetime constraints are not specified here.

[SWS_SoAd_00783] Header list creation for single IF-PDU UDP transmission ⌈For a single IF-PDU UDP transmission request in the separate header and payload buffer path, SoAd shall create exactly one header-list element for that single PDU and provide it to TcpIp. The header-list element shall reference only the serialized SoAd header bytes (Header ID plus payload length). The IF-PDU payload shall either be provided directly via the transmit call payload reference or be provided on TcpIp request via SoAd_CopyTxData().⌋

[SWS_SoAd_00784] Header list entry release at TxConfirmation ⌈For the single IF-PDU UDP transmission path with one header-list element for one IF-PDU, SoAd shall release header-list metadata and serialized-header buffer content after receiving SoAd_TxConfirmation() from a lower layer for the corresponding transmission instance.⌋

[SWS_SoAd_00785] Alignment of SoAd internal transmission buffers ⌈During transmission SoAd shall allocate and maintain its internal transmission buffers (nPduUdpTxBuffer, TP transmit buffers, and header storage) such that their effective alignment is not less than the alignment configured via PduBufferAlignment for the referenced global PDU.⌋

7.2.2 PDU Transmission via IF-API and nPduUdpTxBuffer

[SWS_SoAd_00546]

Upstream requirements: SRS_Eth_00116

⌈In case SoAdTxUdpTriggerMode is set to TRIGGER_NEVER for any PDU route (SoAdPduRouteDest) related to a socket connection and all upper layers belonging to the related socket connection have SoAdTxUpperLayerType set to "IF", SoAd shall use the nPdu feature for this socket connection.⌋

[SWS_SoAd_00547]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and SoAdTxUdpTriggerMode is set to TRIGGER_NEVER for the actual PDU (SoAdPduRouteDest), SoAd shall store the PDU for the socket connection (instead of calling TcpIp_UdpTransmit()).⌋

[SWS_SoAd_00747]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and the related PDU parameter SoAdTxPduCollectionSemantics is set to SOAD_COLLECT_LAST_IS_BEST, SoAd shall only store the transmission request (TxPduId and PduInfoPtr->SduLength of SoAd_IfTransmit()) instead of the PDU data. When SoAd needs to provide the PDU data, SoAd shall retrieve the data from the upper layer by calling <Up>_[SoAd][If]TriggerTransmit().⌋

[SWS_SoAd_00734]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection, the related PDU parameter SoAdTxPduCollectionSemantics is set to SOAD_COLLECT_LAST_IS_BEST and the upper layer doesn't provide all the data, previously requested for transmission according to [SWS_SoAd_00747], via <Up>_[SoAd][If]TriggerTransmit() in the context of SoAd_CopyTxData(), SoAd shall abort the transmission and return BUFREQ_E_NOT_OK.⌋

[SWS_SoAd_00548]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and SoAdTxUdpTriggerMode is set to TRIGGER_ALWAYS for the current PDU (SoAdPduRouteDest) and the resulting PDU data and headers don't exceed SoAdSocketnPduUdpTxBufferMin, SoAd shall transmit all PDUs stored for the socket connection (if any) and the current PDU by calling TcpIp_UdpTransmit().⌋

[SWS_SoAd_00685]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and SoAdTxUdpTriggerMode is set to TRIGGER_ALWAYS for the current PDU (SoAdPduRouteDest) and the resulting PDU data and headers would exceed SoAdSocketnPduUdpTxBufferMin, SoAd shall first transmit all PDUs stored for the socket connection (if any) by calling TcpIp_UdpTransmit() and then the current PDU by calling TcpIp_UdpTransmit() once more.⌋

[SWS_SoAd_00549]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and SoAdTxUdpTriggerMode is set to TRIGGER_NEVER for the current PDU (SoAdPduRouteDest) and the resulting PDU data and headers would exceed SoAdSocketnPduUdpTxBufferMin, SoAd shall first transmit all PDUs stored for the socket connection by calling TcpIp_UdpTransmit() and then store the PDU for the socket connection.⌋

[SWS_SoAd_00690]

Upstream requirements: SRS_Eth_00116

⌈SoAd shall preserve the order of PDUs transmitted via a socket connection that uses the nPdu feature. Pdus collected on the sender side first shall be extracted and indicated to the receivers on the receiving side first as well.⌋

[SWS_SoAd_00691]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and and all PDUs are configured with `SoAdTxPduCollectionSemantics` set to `SOAD_COLLECT_QUEUED`, SoAd shall store all PDUs individually, also PDUs with the same PduId.⌋

[SWS_SoAd_00735]

Upstream requirements: SRS_Eth_00116

⌈In case the nPdu feature is used for a socket connection and all PDUs are configured with `SoAdTxPduCollectionSemantics` set to `SOAD_COLLECT_LAST_IS_BEST`, SoAd shall only store the last instance of each PDU with the same PduId in the sequence their first instances were requested for transmission.⌋

[SWS_SoAd_00736]

Upstream requirements: SRS_Eth_00116

⌈SoAd shall reject configurations in which the transmit properties (see `SoAdTxPduCollectionSemantics`) of the PDUs which are assigned to a socket connection are mixed. Furthermore all socket connnections of a socket connection group shall either be refered solely by PDUs with `SOAD_COLLECT_LAST_IS_BEST` or solely by PDUs with `SOAD_COLLECT_QUEUED` semantic.⌋

[SWS_SoAd_00696]

Upstream requirements: SRS_Eth_00116

⌈SoAd shall maintain a nPdu specific timer for each socket connection using the nPdu feature.⌋

[SWS_SoAd_00550]

Upstream requirements: SRS_Eth_00116

⌈Within `SoAd_MainFunction`() SoAd shall transmit all PDUs stored for a socket connection (if any) by calling `TcpIp_UdpTransmit()` if the nPdu specific timer expired.⌋

[SWS_SoAd_00697]

Upstream requirements: SRS_Eth_00116

⌈If a PDU with `SoAdTxUdpTriggerMode` set to `TRIGGER_NEVER` with a specific `SoAdTxUdpTriggerTimeout` get buffered SoAd shall set the nPdu specific timer to the value of `SoAdTxUdpTriggerTimeout` if the timer is not running or if it is lower than the current nPdu specific timer value.⌋

[SWS_SoAd_00683]

Upstream requirements: SRS_Eth_00116

⌈If a PDU with SoAdTxUdpTriggerMode set to TRIGGER_NEVER without a specific SoAdTxUdpTriggerTimeout get buffered SoAd shall set the the nPdu specific timer to the value of SoAdSocketUdpTriggerTimeout if the timer is not running or if it is lower than the current nPdu specific timer value.⌋

[SWS_SoAd_00684]

Upstream requirements: SRS_Eth_00116

⌈SoAd shall stop the nPdu specific timer when the PDUs stored for the socket connection have been sent⌋

[SWS_SoAd_00737] ⌈For all PDUs that have been stored for a socket connection using the nPdu feature, SoAd shall call the upper layer with the related transmit confirmation function (<Up>_[SoAd][If]TxConfirmation()) within the context of the SoAd_MainFunction() and with result set to

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

⌋

7.2.3 DTLS over UDP

[SWS_SoAd_00795] DTLS connecting: SoCon assignment ⌈Within SoAd_TcpIpEvent() with Event set to TCPIP_UDP_DTLS_STATE_CONNECTING, if no socket connection is already configured with the address in RemoteAddrPtr, SoAd shall choose a socket connection using the best match algorithm according to [SWS_SoAd_00680] and set this socket connection to the remote address specified by RemoteAddrPtr.⌋

[SWS_SoAd_00796] DTLS connecting: no SoCon found ⌈If no socket connection can be found that matches RemoteAddrPtr, SoAd_TcpIpEvent() shall call TcpIp_DtlsClose() for SocketId and RemoteAddrPtr.⌋

[SWS_SoAd_00797] DTLS closed: remote address reset ⌈If SoAd_TcpIpEvent() is called with event TCPIP_UDP_DTLS_STATE_CLOSED, and no upper layer has set the remote address of the related socket connection via SoAd_SetUniqueRemoteAddr() or SoAd_SetRemoteAddr(), SoAd shall reset the remote address of the socket connection to its configured remote address setting.⌋

[SWS_SoAd_00798] DTLS connected: SoCon state update ⌈Within SoAd_TcpIpEvent() called with event TCPIP_UDP_DTLS_STATE_CONNECTED, SoAd shall set the socket connection associated with SocketId and RemoteAddrPtr to state SOAD_SOCON_ONLINE.⌋

[SWS_SoAd_00799] DTLS handshake guard for UDP reception ⌈For a UDP socket connection with SoAdDatagramTlsConnectionRef set, SoAd shall not accept incoming PDUs before DTLS handshake completion and valid remote-address assignment.⌋

Note: Setting the socket connection to SOAD_SOCON_ONLINE triggers the upper-layer state change notification per [SWS_SoAd_00597] and [SWS_SoAd_00741]. The transition to SOAD_SOCON_ONLINE occurs after the DTLS handshake completes; ARP resolution for the remote address is included as part of this transition.

7.2.4 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().⌋

7.2.5 PDU Transmission via TP-API

[SWS_SoAd_00551] ⌈For the transmission of a PDU requested by an upper layer using the TP-API, the SoAd shall

1. Skip further processing and return with E_NOT_OK if the PDU length is 0.

2. Identify the related socket connection and PDU route by using the TxPduId provided at SoAd_TpTransmit().
3. Store the TP transmission request for further processing in the SoAd_MainFunction().

⌋

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

[SWS_SoAd_00552] ⌈In the SoAd_MainFunction() the SoAd shall check for pending TP transmission requests and process a pending request as specified below:

1. Query the available amount of data at the upper layer by calling the configurable callback function <Up>_[SoAd][Tp]CopyTxData() with PduInfoType.SduLength = 0.
2. Depending on the connection type: retrieve data and call the related 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_00786] Separate header and payload buffers for single IF-PDU transmission on non-nPdu socket connections ⌈If PDU header option is enabled (SoAdPduHeaderEnable is TRUE) and IF-PDU transmission is performed on a socket connection where the nPdu feature is not used, SoAd shall apply the following for one transmitted message carrying exactly one IF-PDU:

1. for UDP, SoAd shall call TcpIp_UdpTransmit() using separate header and payload buffers by passing serialized SoAd header bytes via HeaderPtr; payload bytes shall be provided via DataPtr, or via SoAd_CopyTxData() if DataPtr is NULL_PTR. If SoAdKeepLocalPduBuffer is TRUE, SoAd shall set KeepLocalBuffer to TRUE; otherwise SoAd shall set KeepLocalBuffer to FALSE.
2. for TCP, SoAd shall call TcpIp_TcpTransmit() using separate header and payload buffers by passing serialized SoAd header bytes via HeaderPtr; payload bytes shall be provided via SoAd_CopyTxData() as requested by TcpIp.

In this requirement, DataPtr shall carry only payload bytes and shall not include serialized SoAd header bytes.⌋

[SWS_SoAd_00787] TCP transmit handling for non-nPdu socket connections with PDU header enabled ⌈If PDU header option is enabled for transmission over TCP on a socket connection where the nPdu feature is not used, SoAd shall call TcpIp_TcpTransmit() and provide header plus payload bytes using the TcpIp TCP transmit data provisioning path, where SoAd_CopyTxData() is invoked only by TcpIp.⌋

[SWS_SoAd_00788] TotalLength accounting for UDP transmission with separate header and payload buffers ⌈In this UDP separate header and payload buffer path, TotalLength shall represent the complete transmitted byte length (header plus payload) and shall be consistent with the data referenced by HeaderPtr and DataPtr.⌋

[SWS_SoAd_00789] Behavior when PDU header mode is disabled ⌈If SoAdPduHeaderEnable is FALSE for the related UDP socket connection, SoAd shall call TcpIp_UdpTransmit() with HeaderPtr set to NULL_PTR and payload bytes provided via DataPtr and TotalLength.⌋

7.3.1 Scenario coverage: large data transmission via IF API over TCP

[SWS_SoAd_00790] IF TxConfirmation behavior when immediate confirmation is disabled ⌈If SoAdSocketIfImmediateTxConfirmation is FALSE, SoAd shall call the configured transmit confirmation function (<Up>_[SoAd][If]TxConfirmation()) within SoAd_TxConfirmation() after all PDU data of the associated transmission have been confirmed for transmission.⌋

[SWS_SoAd_00791] Immediate IF confirmation option without double confirmation ⌈If SoAdSocketIfImmediateTxConfirmation is TRUE, SoAd shall configure the related TcpIp socket according to [SWS_SoAd_00689] and shall handle IF transmit confirmation as follows:

1. if the transmit request is successfully handed over during request processing in SoAd_MainFunction(), SoAd shall call the configured transmit confirmation function (<Up>_[SoAd][If]TxConfirmation()) immediately after successful handover;
2. if immediate IF transmit confirmation was already issued for the transmission instance, SoAd shall not issue a second IF transmit confirmation when TcpIp later reports transmission completion via SoAd_TxConfirmation().

⌋

[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 SoAdSocketRouteDest configuration

⌋

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

[SWS_SoAd_00563] ⌈In case PDU header option is disabled (SoAdPduHeaderEnable is FALSE) for a socket connection, SoAd shall convert the received message as followed:

1. In case the corresponding SoAdSocketRoute has SoAdRxUpperLayerType = IF, SoAd shall map the UDP message or TCP segment 1:1 to the PDU and process according to [SWS_SoAd_00567].
2. In case the corresponding SoAdSocketRoute has SoAdRxUpperLayerType = TP, SoAd shall map the UDP message or TCP segment to a PDU with unknown length and process according to [SWS_SoAd_00595], [SWS_SoAd_00568] and [SWS_SoAd_00641].

⌋

Note: Since the TCP segmentation of transmitted PDUs is independent of PDU boarders, the received IF PDU can defer from the transmitted 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 UDP socket connection, SoAd shall convert the message into a PDU within SoAd_RxIndication() according to the following:

1. extract the PDU Header from the received message
2. 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 raise the runtime error SOAD_E_INV_PDUHEADER_ID.
3. 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 (1) 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_00771] ⌈In case PDU header option is enabled (SoAdPduHeaderEnable is TRUE) for a TCP socket connection, SoAd shall convert the message into a PDU within SoAd_RxIndication() according to the following:

1. extract remaining PDU payload from the received TCP segment if the previous TCP segment contained fragmented PDU payload. If the end of the TCP segment is not reached continue with (3)
2. extract remaining PDU header from the received TCP segment if the previous TCP segment contained fragmented PDU header and continue with (4).
3. extract the PDU header from the received TCP segment. If the PDU header is fragmented over this and the next TCP segment, then store the received part of the PDU header and continue with (2) for the next TCP segment.
4. select the related socket route according to the received PDU Header ID (SoAd-RxPduHeaderId). If no socket route can be found, simply discard the PDU and raise the runtime error `SOAD_E_INV_PDUHEADER_ID`.
5. 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 (3) until the end of the TCP fragment is reached. If at the end of the TCP segment the PDU header or payload is not completed, it shall be remembered for the next TCP segment.

⌋

[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] ⌈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 runtime error `SOAD_E_NOBUFS`.⌋

[SWS_SoAd_00758]

Upstream requirements: SRS_Eth_00131

⌈If the measurement data is enabled (see SoAdGetAndResetMeasurementDataApi), SoAd shall increment the corresponding measurement data whenever a received PDU is discarded.⌋

[SWS_SoAd_00772]

Status: DRAFT
Upstream requirements: SRS_Eth_00163

⌈If SoAdBswModuleRef is referencing Sd, then SoAd shall process incoming PDUs according to [SWS_SoAd_00773].⌋

[SWS_SoAd_00773]

Status: DRAFT
Upstream requirements: SRS_Eth_00163

⌈In case a received PDU is extracted according to [SWS_SoAd_00559], the matching SoAdSocketRoute references a SoAdRoutingGroup which is also referenced by SdServerServiceActivationRef of SdProvidedMethods and is disabled, then SoAd shall call Sd_RequestRoutingGroupEnable(). Sd_RequestRoutingGroupEnable() provides feedback to Sd that a Sd client tries to use a method of the server.⌋

[SWS_SoAd_00774]

Status: DRAFT
Upstream requirements: SRS_Eth_00163

⌈If Sd_RequestRoutingGroupEnable() returns E_OK, SoAd shall forward the Pdu data to the related upper layer.⌋

Note: Within Sd_RequestRoutingGroupEnable() Sd performs the ACL check. In case of success, it enables the corresponding routing group for faster processing of consecutive PDUs coming from the same remote node and it returns E_OK.

[SWS_SoAd_00775]

Status: DRAFT
Upstream requirements: SRS_Eth_00163

⌈If Sd_RequestRoutingGroupEnable() returns E_NOT_OK, SoAd shall drop the received PDU.⌋

Note: Within Sd_RequestRoutingGroupEnable() Sd performs the ACL check. In case of failure, the corresponding routing group stays disabled and it returns E_NOT_OK.

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

⌋

[SWS_SoAd_00793] IF reception buffer retention with KeepLocalPduBuffer ⌈For IF reception with KeepLocalPduBuffer set to TRUE, if SoAd_RxIndication() is invoked with KeepLocalBuffer set to TRUE, SoAd shall provide the received data to the upper layer via <Up>_[SoAd][If]RxIndication() and keep the source-related local buffer valid until the consumer invokes SoAd_ReleaseRxBuffer() for the related PDU. After SoAd_ReleaseRxBuffer() is invoked, SoAd shall release the related lower-layer receive buffer via TcpIp_ReleaseRxBuffer.⌋

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

[SWS_SoAd_00740]

Upstream requirements: SRS_Eth_00124

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

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

[SWS_SoAd_00568] ⌈SoAd shall perform the following further actions within the SoAd_RxIndication() function for reception of a PDU from a socket connection with PDU Header mode disabled to an upper layer using the TP-API:

1. if the SoAd receive buffer does not contain any TP data for this socket connection

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

(b) If not all data can be processed (i.e. forwarded to an upper layer or stored in a SoAd receive buffer), discard all received data and skip further processing.

(c) Copy all received data which can be accepted by the upper layer module determined at (a) to the upper layer by calling <Up>_[SoAd][Tp]CopyRxData().

(d) Copy all remaining data (i.e. data which are received but not delivered to the upper layer) to a SoAd receive buffer for later processing by SoAd_MainFunction().

2. if the SoAd receive buffer already contains TP data for this socket connection and is able to store all (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 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() with E_NOT_OK in case the socket connection is disconnected while an active TP reception.

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

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

1. if no TP reception is in progress for the related socket connection
   (a) After reception of a complete PDU Header, call <Up>_[SoAd][Tp]StartOfReception() of the related upper layer module with RxPduId set to the ID specified by the upper layer module for the PDU referenced by SoAdRxPduRef, set TpSduLength to the length specified in the PDU Header, and set PduInfoType.SduDataPtr and PduInfoType.SduLength to provide already received PDU data to the upper layer.
   (b) if not all data can be processed (i.e. forwarded to an upper layer or stored in a SoAd receive buffer), discard all received data, call <Up>_[SoAd][Tp]RxIndication() with the same RxPduId as used at <Up>_[SoAd][Tp]StartOfReception() and result set to E_NOT_OK and skip further processing
   (c) call <Up>_[SoAd][Tp]CopyRxData() of the related upper layer module with the same RxPduId as used at <Up>_[SoAd][Tp]StartOfReception() and PduInfoType.SduDataPtr pointing to the PDU data provided by SoAd_RxIndication() and PduInfoType.SduLength set to minimum of the received PDU data and the available receive buffer in the upper layer module specified by bufferSizePtr at <Up>_[SoAd][Tp]StartOfReception())
   (d) call <Up>_[SoAd][Tp]RxIndication() if the complete PDU has been forwarded to the upper layer, otherwise copy all received data which could not be forwarded to the upper layer to a SoAd receive buffer for later processing by SoAd_MainFunction()
2. if a TP reception is in progress for the related socket connection and the related SoAd receive buffer is able to store all received data: copy all received data to the related SoAd receive buffer for later processing by SoAd_MainFunction()

⌋

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

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

[SWS_SoAd_00573] ⌈If <Up>_[SoAd][Tp]CopyRxData() does not return BUFREQ_OK for a socket connection with PDU Header mode enabled, SoAd shall

terminate the TP receive session, simply discard all data of the PDU and call <Up>_[SoAd][Tp]RxIndication() with E_NOT_OK.⌋

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

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

⌋

7.5 Best Match Algorithm

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

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

⌋

7.6 Message Acceptance Policy

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

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

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

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

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

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

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

7.7 TP PDU 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]

Upstream requirements: SRS_Eth_00085

⌈If a UDP socket connection is configured with a SoAdSocketUdpAliveSupervisionTimeout and the remote address was overwritten, as described in [SWS_SoAd_00592], the alive supervision timer for this socket connection shall be started with the value specified by the configuration parameter SoAdSocketUdpAliveSupervisionTimeout.⌋

[SWS_SoAd_00742]

Upstream requirements: SRS_Eth_00085

⌈If a UDP socket connection is configured with a SoAdSocketUdpAliveSupervisionTimeout and a datagram is received that passes the message acceptance filter, the timer shall be restarted with the value specified by the configuration parameter SoAdSocketUdpAliveSupervisionTimeout.⌋

[SWS_SoAd_00695]

Upstream requirements: SRS_Eth_00085

⌈If a UDP socket connection is configured with a SoAdSocketUdpAliveSupervisionTimeout, the alive supervision timer runs out and the remote address was not set by the upper layer, SoAd shall

1. change the state of the socket connection to SOAD_SOCON_RECONNECT,
2. deactivate the alive supervision timer and
3. reset the remote address to the configured remote address at SoAd_MainFunction().

⌋

[SWS_SoAd_00762]

Upstream requirements: SRS_Eth_00085

⌈If the function SoAd_ReleaseRemoteAddr() resets the remote address of a socket connection to a configured remote address containing wildcards and socket connection is in mode SOAD_SOCON_ONLINE, SoAd shall change the mode of the socket connection to SOAD_SOCON_RECONNECT.⌋

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 SoAdSocketConnectionGroup SoAd shall maintain independent states for each SocketConnection that is part of the referenced SoAdSocketConnectionGroup and handle them as if they were separate RoutingGroups.⌋

[SWS_SoAd_00560] ⌈If SoAd_IfTransmit() is called with TxPduId specifying a SoAdPduRouteDest which belongs only to inactive RoutingGroups, SoAd shall always skip the transmission for this SoAdPduRouteDest and shall consider the transmission as successful unless all SoAdPduRouteDest of a SoAdPduRoute belong only to inactive RoutingGroups. In the latter case SoAd shall return E_NOT_OK.⌋

[SWS_SoAd_00561] ⌈If SoAd_TpTransmit() is called with TxPduId specifying a SoAdPduRouteDest which belongs only to inactive RoutingGroups, SoAd shall always skip the transmission for this SoAdPduRouteDest and shall consider the transmission as successful unless all SoAdPduRouteDest of a SoAdPduRoute belong only to inactive RoutingGroups. In the latter case SoAd shall return E_NOT_OK.⌋

[SWS_SoAd_00600] ⌈If a PDU is received according to SoAdSocketRouteDest which belongs only to inactive RoutingGroups, SoAd shall simply discard the PDU.⌋

[SWS_SoAd_00760] ⌈For SoAdRoutingGroups that are referenced by a SoAdSocketRouteDest that belongs to a SoAdSocketRoute referring to a SoAdSocketConnectionGroup, SoAd shall maintain independent states for each socket connection that is part of the referenced SoAdSocketConnectionGroup and handle them as if they were separate SoAdRoutingGroups.⌋

[SWS_SoAd_00761] ⌈If SoAd_EnableSpecificRouting() is called with SoConId belonging to a SoAdSocketConnectionGroup and for a SoAdRoutingGroup that is referenced by a SoAdSocketRouteDest that belongs to a SoAdSocketRoute referring to the same SoAdSocketConnectionGroup and that contains more than one SoAdSocketRouteDest, SoAd shall only enable the independent state for the socket connection if:

1. the same SoAdSocketRouteDest is not active for any other socket connection of the same SoAdSocketConnectionGroup (i.e. SoAdSocketRouteDest is not already mapped to another socket connection) and
2. no SoAdRoutingGroup is enabled on the socket connection which is referenced by another SoAdSocketRouteDest of the same SoAdSocketRoute (i.e. the socket connection is not mapped to another SoAdSocketRouteDest of the same SoAdSocketRoute).

Otherwise, E_NOT_OK shall be returned.⌋

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

7.10 PDU fan-out

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

[SWS_SoAd_00722] ⌈SoAd shall handle SoAdPduRoute with SoAdPduRouteDest referring to a SoAdSocketConnectionGroup as if they were separate SoAdPduRouteDest referring to each SocketConnection of this Group.⌋

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

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

7.11 Buffer handling

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

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

7.12 Security Events

[SWS_SoAd_00763] ⌈If security event reporting has been enabled for the SoAd module (SoAdEnableSecurityEventReporting = true) the respective security events shall be reported to the IdsM via the interfaces defined in AUTOSAR_SWS_BSWGeneral [2].⌋

The following table lists the security events which are standardized for the SoAd module together with their trigger conditions:

[SWS_SoAd_00764] Security events for SoAd ⌈

Name	*Description*	*ID*
SEV_DROP_PDU_RX_UDP	SoAd dropped a PDU. The PDU violates stack configuration and was received via a UDP socket.	6
SEV_DROP_MSG_RX_UDP_LENGTH	SoAd dropped a message. The message contains at least one PDU which violates stack configuration and was received via a UDP socket. The violation relates to the length of the PDUs compared to the overall length of the message.	7
SEV_DROP_MSG_RX_UDP_SOCKET	SoAd received a UDP message which violates stack configuration and was dropped. No suitable socket connection matching to configuration was found.	8
SEV_REJECTED_TCP_CONNECTION	SoAd rejected a TCP connection. The connection request violates stack configuration.	9

▽

△

Name	*Description*	*ID*
SEV_DROP_PDU_RX_TCP	SoAd dropped a PDU. The PDU violates stack configuration and was received via a TCP socket.	50

⌋

Context data is not provided by the SoAd module for the security events.

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 [2, General Specification of Basic Software Modules] 7.2 *"Error Handling"*.

7.13.1 Development Errors

[SWS_SoAd_00777] Development error reporting for pointers ⌈If development error detection is enabled each API of the SoAd shall check the validity of pointer arguments. If the check fails, the API shall raise the development error `SOAD_E_PARAM_POINTER`.⌋

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

[SWS_SoAd_00101] Definition of development errors in module SoAd ⌈

Type of error	*Related error code*	*Error value*
API service called before initializing the module	SOAD_E_UNINIT	0x01
API service called with NULL pointer	SOAD_E_PARAM_POINTER	0x02
Invalid argument	SOAD_E_INV_ARG	0x03
Invalid PDU ID	SOAD_E_INV_PDUID	0x06
Invalid socket address	SOAD_E_INV_SOCKETID	0x07
Invalid configuration set selection	SOAD_E_INIT_FAILED	0x08
Invalid meta data	SOAD_E_INV_METADATA	0x09

⌋

7.13.2 Runtime Errors

[SWS_SoAd_00794] Runtime error reporting for lost transmission message ⌈If SoAd detects loss of a transmission message after request acceptance (for example, `SoAd_TxConfirmation`() with `Length` = 0 or equivalent internal loss such as asynchronous-copy failure), SoAd shall report runtime error `SOAD_E_MESSAGE_LOST`.⌋

The following table lists runtime error IDs the SoAd shall use for reporting of runtime errors to the Default Error Tracer:

[SWS_SoAd_00759] Definition of runtime errors in module SoAd ⌈

Type of error	*Related error code*	*Error value*
No buffer space available	SOAD_E_NOBUFS	0x04
Unknown PduHeader ID	SOAD_E_INV_PDUHEADER_ID	0x05
Automatic TCP connection failed	SOAD_E_TCP_AUTOCONNECT_FAILED	0x10

⌋

7.13.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

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

[SWS_SoAd_00503] Definition of imported datatypes of module SoAd ⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	ListElemStructType (draft)
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TPParameterType
	ComStack_Types.h	TpDataStateType
IdsM	IdsM.h	IdsM_SecurityEventIdType
	Rte_IdsM_Type.h	IdsM_TimestampDataType
Std	Std_Types.h	Std_ReturnType
TcpIp	TcpIp.h	TcpIp_DomainType
	TcpIp.h	TcpIp_EventType
	TcpIp.h	TcpIp_IpAddrAssignmentType
	TcpIp.h	TcpIp_IpAddrStateType
	TcpIp.h	TcpIp_LocalAddrIdType
	TcpIp.h	TcpIp_ParamIdType
	TcpIp.h	TcpIp_ProtocolType
	TcpIp.h	TcpIp_ReturnType
	TcpIp.h	TcpIp_SockAddrType
	TcpIp.h	TcpIp_SocketIdType
	TcpIp.h	TcpIp_StateType

⌋

8.2 Type definitions

8.2.1 SoAd_SoConIdType

[SWS_SoAd_00518] Definition of datatype SoAd_SoConIdType ⌈

Name	SoAd_SoConIdType	
Kind	Type	
Derived from	*Basetype*	*Variation*
	uint16	–
	uint8	–

▽

△

Range	0..<SoAdSoConMax>	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.		
Available via	SoAd.h		

⌋

8.2.2 SoAd_SoConModeType

[SWS_SoAd_00512] Definition of datatype SoAd_SoConModeType ⌈

Name	SoAd_SoConModeType		
Kind	Enumeration		
Range	SOAD_SOCON_ONLINE	–	–
	SOAD_SOCON_ RECONNECT	–	–
	SOAD_SOCON_OFFLINE	–	–
Description	type to specify the state of a SoAd socket connection.		
Available via	SoAd.h		

⌋

8.2.3 SoAd_RoutingGroupIdType

[SWS_SoAd_00519] Definition of datatype SoAd_RoutingGroupIdType ⌈

Name	SoAd_RoutingGroupIdType		
Kind	Type		
Derived from	***Basetype***	***Variation***	
	uint16	–	
	uint8	–	
Range	0..<SoAdRoutingGroup Max>	0..<SoAdRoutingGroup Max>	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.		
Available via	SoAd.h		

⌋

8.2.4 SoAd_ConfigType

[SWS_SoAd_00210] Definition of datatype SoAd_ConfigType ⌈

Name	SoAd_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	The content of the configuration data structure is implementation specific.
Description	Configuration data structure of the SoAd module.	
Available via	SoAd.h	

⌋

8.2.5 SoAd_MeasurementIdxType

[SWS_SoAd_91010] Definition of datatype SoAd_MeasurementIdxType ⌈

Name	SoAd_MeasurementIdxType		
Kind	Type		
Derived from	uint8		
Range	SOAD_MEAS_DROP_TCP	0x01	Measurement index of dropped PDUs caused by invalid destination TCP-Port
	SOAD_MEAS_DROP_UDP	0x02	Measurement index of dropped PDUs caused by invalid destination UDP-Port
	SOAD_MEAS_ RESERVED_1	0x03-0x7F	reserved by AUTOSAR
	SOAD_MEAS_ RESERVED_2	0x80-0xEF	Vendor specific range
	SOAD_MEAS_ RESERVED_3	0xF0-0xFE	reserved by AUTOSAR (future use)
	SOAD_MEAS_ALL	0xFF	represents all measurement indexes
Description	Index to select specific measurement data		
Available via	SoAd.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 General

8.3.1.1 SoAd_Init

[SWS_SoAd_00093] Definition of API function SoAd_Init ⌈

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.	
Available via	SoAd.h	

⌋

[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] Definition of API function SoAd_IfTransmit ⌈

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

▽

△

Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `E_NOT_OK`: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	SoAd.h	

▽

[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_UNINIT`.⌋

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

[SWS_SoAd_00732]

Upstream requirements: SRS_BSW_00337

⌈SoAd shall only consider PduInfoPtr->SduDataPtr set to `NULL_PTR` as valid if SoAdIfTriggerTransmit is set to `TRUE` for the respective upper layer.⌋

[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 identified by `TxPduId`.⌋

Note: An IF transmission is considered as ongoing until `SoAd_IfTransmit`() returns. A TP transmission is considered as ongoing until SoAd calls `<Up>_SoAdTpTxConfirmation()`.

8.3.2.2 SoAd_IfRoutingGroupTransmit

[SWS_SoAd_00656] Definition of API function SoAd_IfRoutingGroupTransmit ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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] Definition of API function SoAd_IfSpecificRoutingGroup Transmit ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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] Definition of API function SoAd_TpTransmit ⌈

Service Name	SoAd_TpTransmit	
Syntax	`Std_ReturnType SoAd_TpTransmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x53	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

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

Note: No TxConfirmation is required when SoAd_TpTransmit() failed.

8.3.2.5 SoAd_ReleaseRxBuffer

[SWS_SoAd_91012] Definition of API function SoAd_ReleaseRxBuffer ⌈

Service Name	SoAd_ReleaseRxBuffer	
Syntax	`void SoAd_ReleaseRxBuffer ( PduIdType RxPduId )`	
Service ID [hex]	0x56	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId	
Parameters (in)	RxPduId	Identifier of the received PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication from the upper layer to release the lower layer reception buffer.	
Available via	SoAd.h	

⌋

8.3.3 Transmit/Receive Cancelation API

8.3.3.1 SoAd_TpCancelTransmit

[SWS_SoAd_00522] Definition of API function SoAd_TpCancelTransmit ⌈

Service Name	SoAd_TpCancelTransmit	
Syntax	`Std_ReturnType SoAd_TpCancelTransmit ( PduIdType TxPduId )`	
Service ID [hex]	0x54	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Cancellation was executed successfully by the destination module. `E_NOT_OK`: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

[SWS_SoAd_00606] ⌈If development error detection is enabled: SoAd_TpCancelTransmit() shall check parameter TxPduId for being valid. If the check fails, SoAd_TpCancelTransmit() shall raise the development error SOAD_E_INV_PDUID.⌋

8.3.3.2 SoAd_TpCancelReceive

[SWS_SoAd_00521] Definition of API function SoAd_TpCancelReceive ⌈

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

⌋

[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_UNINIT.⌋

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

8.3.4 Information and Control API

8.3.4.1 SoAd_GetSoConId

[SWS_SoAd_00509] Definition of API function SoAd_GetSoConId ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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] Definition of API function SoAd_OpenSoCon ⌈

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.	

▽

△

Available via	SoAd.h

⌋

[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_UNINIT.⌋

[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] Definition of API function SoAd_CloseSoCon ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

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

8.3.4.4 SoAd_GetSoConMode

[SWS_SoAd_91001] Definition of API function SoAd_GetSoConMode ⌈

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

⌋

8.3.4.5 SoAd_RequestIpAddrAssignment

[SWS_SoAd_00520] Definition of API function SoAd_RequestIpAddrAssignment ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

8.3.4.6 SoAd_ReleaseIpAddrAssignment

[SWS_SoAd_00536] Definition of API function SoAd_ReleaseIpAddrAssignment
⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[SWS_SoAd_00619] ⌈If development error detection is enabled: SoAd_ReleaseIpAddrAssignment() shall check parameter SoConId for being valid. If the check fails, SoAd_ReleaseIpAddrAssignment() shall raise the development error SOAD_E_INV_ARG.⌋

8.3.4.7 SoAd_GetLocalAddr

[SWS_SoAd_00506] Definition of API function SoAd_GetLocalAddr ⌈

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
Available via	SoAd.h

⌋

[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_UNINIT.⌋

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

8.3.4.8 SoAd_GetPhysAddr

[SWS_SoAd_00507] Definition of API function SoAd_GetPhysAddr ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

8.3.4.9 SoAd_GetRemoteAddr

[SWS_SoAd_00655] Definition of API function SoAd_GetRemoteAddr ⌈

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	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

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

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

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

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

8.3.4.10 SoAd_EnableRouting

[SWS_SoAd_00516] Definition of API function SoAd_EnableRouting ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

8.3.4.11 SoAd_EnableSpecificRouting

[SWS_SoAd_00714] Definition of API function SoAd_EnableSpecificRouting ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

8.3.4.12 SoAd_DisableRouting

[SWS_SoAd_00517] Definition of API function SoAd_DisableRouting ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[SWS_SoAd_00626] ⌈If development error detection is enabled: SoAd_Disable-Routing() shall check parameter id for being valid. If the check fails, SoAd_Disable-Routing() shall raise the development error SOAD_E_INV_ARG.⌋

8.3.4.13 SoAd_DisableSpecificRouting

[SWS_SoAd_00717] Definition of API function SoAd_DisableSpecificRouting ⌈

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.	
Available via	SoAd.h	

⌋

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

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

8.3.4.14 SoAd_SetRemoteAddr

[SWS_SoAd_00515] Definition of API function SoAd_SetRemoteAddr ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

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

[SWS_SoAd_00532]

Upstream requirements: SRS_Eth_00085

⌈The function SoAd_SetRemoteAddr() shall only proceed if SoConId refers to a socket connection which is not currently locked by the SoAd. If it is locked, the request shall be rejected and E_NOT_OK shall be returned.⌋

[SWS_SoAd_00533] ⌈The function SoAd_SetRemoteAddr() shall set the remote address of the socket connection referred by parameter SoConId according to the IP address and port specified by parameter RemoteAddrPtr.⌋

[SWS_SoAd_00687] ⌈If the function SoAd_SetRemoteAddr() is used to set the remote address of a socket connection that is in the mode SOAD_SOCON_ONLINE to a value that contains wildcards, SoAd shall change the mode of the socket connection to SOAD_SOCON_RECONNECT.⌋

[SWS_SoAd_00699] ⌈SoAd_SetRemoteAddr() shall refuse the request if the domain set in RemoteAddrPtr does not match the TcpIp_DomainType of the local address related to the socket connection identified by SoConId and return E_NOT_OK. If development error detection is enabled, the service SoAd_SetRemoteAddr() shall also raise the development error SOAD_E_INV_ARG.⌋

8.3.4.15 SoAd_SetUniqueRemoteAddr

[SWS_SoAd_00671] Definition of API function SoAd_SetUniqueRemoteAddr ⌈

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 SoAdSocket ConnectionGroup.
	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 SoAdSocketConnection Group 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.	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

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

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

[SWS_SoAd_00676] ⌈If no socket connection is already configured with the address specified by RemoteAddrPtr, `SoAd_SetUniqueRemoteAddr`() shall:

1. choose an unused socket connection using the best match algorithm described in [SWS_SoAd_00680]
2. set it to the remote address specified by RemoteAddrPtr
3. set `AssignedSoConIdPtr` to the index of the chosen socket connection and

4. return E_OK.

A socket connection is "unused" if its actual remote address has an IP address wildcard and/or port wildcard.⌋

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

[SWS_SoAd_00700] ⌈SoAd_SetUniqueRemoteAddr() shall refuse the request if the domain set in RemoteAddrPtr does not match the 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.⌋

8.3.4.16 SoAd_ReleaseRemoteAddr

[SWS_SoAd_00733] Definition of API function SoAd_ReleaseRemoteAddr ⌈

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

⌋

[SWS_SoAd_00744]

Upstream requirements: SRS_BSW_00337

⌈If development error detection is enabled and SoConId refers to a socket connection with configuration parameter SoAdSocketAutomaticSoConSetup set to TRUE, the function SoAd_ReleaseRemoteAddr() shall raise the development error SOAD_E_INV_ARG.⌋

[SWS_SoAd_00745]

Upstream requirements: SRS_Eth_00085

⌈The function SoAd_ReleaseRemoteAddr() shall only immediately proceed if the SoConId refers to a socket connection which is not currently locked by the SoAd. If it

is locked, the request shall be postponed to the SoAd_MainFunction() and executed once the lock is released.⌋

[SWS_SoAd_00746]

Upstream requirements: SRS_Eth_00085

⌈The function SoAd_ReleaseRemoteAddr() shall reset the remote address of the socket connection referred by parameter SoConId to the configured remote address setting.⌋

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

8.3.4.17 SoAd_TpChangeParameter

[SWS_SoAd_00508] Definition of API function SoAd_TpChangeParameter ⌈

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

⌋

[SWS_SoAd_00730] ⌈SoAd_TpChangeParameter() shall always reject requests by returning E_NOT_OK.⌋

8.3.4.18 SoAd_ReadDhcpHostNameOption

[SWS_SoAd_00681] Definition of API function SoAd_ReadDhcpHostNameOption
⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

[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 [9], 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 [10], if the socket connection identified by `SoConId` is related to a local address of the `TcpIp_DomainType` `TCPIP_AF_INET6`.⌋

8.3.4.19 SoAd_WriteDhcpHostNameOption

[SWS_SoAd_00679] Definition of API function SoAd_WriteDhcpHostNameOption
⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

[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 [9], 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 [10], if the socket connection identified by `SoConId` is related to a local address of the `TcpIp_DomainType` `TCPIP_AF_INET6`.⌋

8.3.4.20 SoAd_GetAndResetMeasurementData

[SWS_SoAd_00010] Definition of API function SoAd_GetAndResetMeasurement Data ⌈

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

⌋

[SWS_SoAd_00757]

Upstream requirements: SRS_Eth_00131

⌈The function `SoAd_GetAndResetMeasurementData`() shall be pre compile time configurable On/Off by the configuration parameter: `SoAdGetAndReset-MeasurementDataApi`.⌋

[SWS_SoAd_00748]

Upstream requirements: SRS_Eth_00131

⌈`SoAd_GetAndResetMeasurementData`() shall return measurement data for selected measurement index.⌋

[SWS_SoAd_00749]

Upstream requirements: SRS_Eth_00131

⌈For measurement index `SOAD_MEAS_DROP_TCP` `SoAd_GetAndReset-MeasurementData`() shall return the number of dropped TCP-Port PDUs.⌋

[SWS_SoAd_00750]

Upstream requirements: SRS_Eth_00131

⌈For measurement index `SOAD_MEAS_DROP_UDP` `SoAd_GetAndReset-MeasurementData`() shall return the number of dropped UDP-Port PDUs.⌋

[SWS_SoAd_00751]

Upstream requirements: SRS_Eth_00131

⌈SoAd_GetAndResetMeasurementData() shall return E_NOT_OK if the requested measurement index is not supported.⌋

[SWS_SoAd_00752]

Upstream requirements: SRS_Eth_00131

⌈SoAd_GetAndResetMeasurementData() shall additionally reset the measurement data to 0 if the MeasurementResetNeeded is true. The reset shall be applied after measurement data has been read.⌋

[SWS_SoAd_00753]

Upstream requirements: SRS_Eth_00131

⌈SoAd_GetAndResetMeasurementData() shall reset all existing measurement data to 0, if MeasurementResetNeeded is true and measurement index is set to SOAD_MEAS_ALL.⌋

[SWS_SoAd_00754]

Upstream requirements: SRS_Eth_00131

⌈All measurement data which counts data shall not overrun.⌋

[SWS_SoAd_00755]

Upstream requirements: SRS_Eth_00131

⌈SoAd_GetAndResetMeasurementData() shall accept MeasurementDataPtr set to NULL_PTR. In this case the measurement data shall not be copied.⌋

[SWS_SoAd_00756]

Upstream requirements: SRS_Eth_00131

⌈If development error detection is enabled: SoAd_GetAndResetMeasurement-Data() shall check that the service SoAd_Init() was previously called. If the check fails, SoAd_GetAndResetMeasurementData() shall raise the development error SOAD_E_UNINIT.⌋

8.3.4.21 SoAd_IsConnectionReady

[SWS_SoAd_91011] Definition of API function SoAd_IsConnectionReady ⌈

Service Name	SoAd_IsConnectionReady	
Syntax	`TcpIp_ReturnType SoAd_IsConnectionReady ( SoAd_SoConIdType SoConId, const TcpIp_SockAddrType* RemoteAddrPtr )`	
Service ID [hex]	0x55	
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 the request.
	RemoteAddrPtr	Pointer to the structure containing the requested remote IP address and port.
Parameters (inout)	None	
Parameters (out)	None	
Return value	TcpIp_ReturnType	`TCPIP_E_OK` - Connection is ready for communication. `TCPIP_E_NOT_OK` - Request was rejected. `TCPIP_E_PENDING` - Connection establishment in progress.
Description	API allows to check if a communication over this socket connection is possible for a dedicated remote address. It includes that the socket connection is bound to a socket, a physical address is available for the requested remote address and if a security association is configured that a secured connection is already established.	
Available via	SoAd.h	

⌋

[SWS_SoAd_00770] ⌈If development error detection is enabled: `SoAd_IsConnectionReady`() shall check parameter `SoConId` for being valid. If the check fails, the function shall raise the development error `SOAD_E_INV_ARG`.⌋

8.3.5 TCP/IP Stack

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

8.3.5.1 SoAd_RxIndication

[SWS_SoAd_00097] Definition of API function SoAd_RxIndication ⌈

Service Name	SoAd_RxIndication
Syntax	`void SoAd_RxIndication ( TcpIp_SocketIdType SocketId, const TcpIp_SockAddrType* RemoteAddrPtr, const uint8* BufPtr, uint16 Length, boolean KeepLocalBuffer )`
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.
	KeepLocalBuffer	TRUE: Call TcpIp_ReleaseRxBuffer to free the provided buffer. FALSE: Buffer is freed upon return of the API.
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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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.⌋

8.3.5.2 SoAd_CopyTxData

[SWS_SoAd_00523] Definition of API function SoAd_CopyTxData ⌈

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.
	BufLength	Length of provided data buffer.
Parameters (inout)	None	

▽

△

Parameters (out)	BufPtr	Pointer to buffer for transmission data.
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 Tcp Ip_Transmit().	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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.⌋

8.3.5.3 SoAd_TxConfirmation

[SWS_SoAd_00098] Definition of API function SoAd_TxConfirmation ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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.⌋

8.3.5.4 SoAd_TcpAccepted

[SWS_SoAd_00099] Definition of callback function SoAd_TcpAccepted ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT.⌋

[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.⌋

8.3.5.5 SoAd_TcpConnected

[SWS_SoAd_00100] Definition of callback function SoAd_TcpConnected ⌈

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.	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

[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`.⌋

8.3.5.6 SoAd_TcpIpEvent

[SWS_SoAd_00146] Definition of callback function SoAd_TcpIpEvent ⌈

Service Name	SoAd_TcpIpEvent	
Syntax	`void SoAd_TcpIpEvent ( TcpIp_SocketIdType SocketId, TcpIp_EventType Event, const TcpIp_SockAddrType* RemoteAddrPtr )`	
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.
	RemoteAddrPtr	IP address and port of the remote host.

▽

△

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.
Available via	SoAd.h

⌋

[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_UNINIT.⌋

[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.⌋

[SWS_SoAd_00800] DTLS DET: domain mismatch detection ⌈If development error detection is enabled and SoAd_TcpIpEvent() is called with event TCPIP_UDP_DTLS_STATE_CONNECTING, TCPIP_UDP_DTLS_STATE_CONNECTED, or TCPIP_UDP_DTLS_STATE_CLOSED, and with a domain in RemoteAddrPtr that does not match TcpIp_DomainType of the local address related to the socket identified by SocketId, SoAd shall raise SOAD_E_INV_ARG.⌋

[SWS_SoAd_00801] DTLS DET: NULL RemoteAddrPtr ⌈If development error detection is enabled and SoAd_TcpIpEvent() is called with event TCPIP_UDP_DTLS_STATE_CONNECTING, TCPIP_UDP_DTLS_STATE_CONNECTED, or TCPIP_UDP_DTLS_STATE_CLOSED, and RemoteAddrPtr set to NULL_PTR, SoAd shall raise SOAD_E_PARAM_POINTER.⌋

[SWS_SoAd_00803] DTLS DET: invalid domain value ⌈If development error detection is enabled and SoAd_TcpIpEvent() is called with event TCPIP_UDP_DTLS_STATE_CONNECTING, TCPIP_UDP_DTLS_STATE_CONNECTED, or TCPIP_UDP_DTLS_STATE_CLOSED, and with the domain field of RemoteAddrPtr set to a value that is neither TCPIP_AF_INET nor TCPIP_AF_INET6, SoAd shall raise SOAD_E_INV_ARG.⌋

Note: A domain mismatch against the local socket domain is handled separately by the preceding requirement. This requirement covers the distinct case where RemoteAddrPtr carries a value outside the valid TcpIp_DomainType range (TCPIP_AF_INET and TCPIP_AF_INET6 are the only defined values).

8.3.5.7 SoAd_LocalIpAddrAssignmentChg

[SWS_SoAd_00209] Definition of callback function SoAd_LocalIpAddrAssignmentChg ⌈

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).	
Available via	SoAd.h	

⌋

[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_UNINIT`.⌋

[SWS_SoAd_00729] ⌈If `SoAd_LocalIpAddrAssignmentChg`() is called with the parameter `IpAddrId` set to a local address which is not referenced by any `SoAdSocketConnectionGroup`, SoAd shall ignore the notification and return without any further action.⌋

8.4 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.

8.4.1 Terms and definitions

For details refer to [2] Chapter 8.4 *"Scheduled functions"*.

8.4.2 SoAd_MainFunction

[SWS_SoAd_00121] Definition of scheduled function SoAd_MainFunction ⌈

Service Name	SoAd_MainFunction
Syntax	`void SoAd_MainFunction ( void )`
Service ID [hex]	0x19
Description	Schedules the Socket Adaptor. (Entry point for scheduling)
Available via	SchM_SoAd.h

⌋

[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 `SoAdMainFunctionPeriod`.⌋

8.5 Expected interfaces

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

8.5.1 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] Definition of mandatory interfaces required by module So Ad ⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
TcpIp_<Up>GetSocket	TcpIp.h	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_Bind	TcpIp.h	By this API service the TCP/IP stack is requested to bind a UDP or TCP socket to a local resource.
TcpIp_ChangeParameter	TcpIp.h	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	TcpIp.h	By this API service the TCP/IP stack is requested to close the socket and release all related resources.
TcpIp_GetCtrlIdx	TcpIp.h	TcpIp_GetCtrlIdx returns the index of the controller related to LocalAddrId.

▽

△

API Function	Header File	Description
TcpIp_GetIpAddr	TcpIp.h	Obtains the local IP address actually used by LocalAddrId, the netmask and default router
TcpIp_GetPhysAddr	TcpIp.h	Obtains the physical source address used by the EthIf controller implicitly specified via LocalAddrId.
TcpIp_GetRemotePhysAddr	TcpIp.h	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_IsConnectionReady	TcpIp.h	API allows to check if a communication over this socket is possible for a dedicated remote address. It includes that the socket is bound, a physical address is available for the requested remote address and if a security association is configured that a secured connection is already established.
TcpIp_ReleaseIpAddrAssignment	TcpIp.h	By this API service the local IP address assignment for the IP address specified by LocalAddrId shall be released.
TcpIp_RequestComMode	TcpIp.h	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	TcpIp.h	By this API service the local IP address assignment for the IP address specified by LocalAddrId shall be initiated.
TcpIp_TcpConnect	TcpIp.h	By this API service the TCP/IP stack is requested to establish a TCP connection to the configured peer.
TcpIp_TcpListen	TcpIp.h	By this API service the TCP/IP stack is requested to listen on the TCP socket specified by the socket identifier.
TcpIp_TcpReceived	TcpIp.h	By this API service the reception of socket data is confirmed to the TCP/IP stack.
TcpIp_TcpTransmit	TcpIp.h	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	TcpIp.h	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.

⌋

8.5.2 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] Definition of optional interfaces requested by module SoAd
⌈

API Function	*Header File*	*Description*
Det_ReportError	Det.h	Service to report development errors.
IdsM_ReportSecurityEvent	IdsM.h	This API is the interface to report security events to the IdsM.
PduR_SoAdTpCopyRxData	PduR_SoAd.h	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 buffer is written to the position indicated by bufferSizePtr.
PduR_SoAdTpCopyTxData	PduR_SoAd.h	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.
Sd_RequestRoutingGroupEnable (draft)	Sd.h	Callback function, which will be provided to SoAd to be able to trigger the ACL policy check or explicitly grant access on the received Method call request **Tags:** atp.Status=draft
TcpIp_DhcpReadOption	TcpIp.h	By this API service the TCP/IP stack retrieves DHCP option data identified by parameter option for already received DHCP options.
TcpIp_DhcpV6ReadOption	TcpIp.h	By this API service the TCP/IP stack retrieves DHCPv6 option data identified by parameter option for already received DHCPv6 options.
TcpIp_DhcpV6WriteOption	TcpIp.h	By this API service the TCP/IP stack writes the DHCPv6 option data identified by parameter option.
TcpIp_DhcpWriteOption	TcpIp.h	By this API service the TCP/IP stack writes the DHCP option data identified by parameter option.
TcpIp_ReleaseRxBuffer	TcpIp.h	Indication from the upper layer to release the lower layer reception buffer for the oldest PDU.

⌋

8.5.3 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.

8.5.3.1 <Up>_[SoAd][If]RxIndication

[SWS_SoAd_00106] Definition of configurable interface <Up>_[SoAd][If]RxIndication ⌈

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 PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication of a received PDU from a lower layer communication interface module.	
Available via	<none>	

⌋

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

[SWS_SoAd_00663] Definition of configurable interface <Up>_[SoAd][If]TriggerTransmit ⌈

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 data 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.	
Available via	<none>	

⌋

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

[SWS_SoAd_00107] Definition of configurable interface <Up>_[SoAd][If]TxConfirmation ⌈

Service Name	<Up>_[SoAd][If]TxConfirmation	

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

Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	`E_OK`: The PDU was transmitted successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	<none>	

⌋

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

[SWS_SoAd_00138] Definition of configurable interface <Up>_[SoAd][Tp]StartOfReception ⌈

Service Name	<Up>_[SoAd][Tp]StartOfReception	

Syntax	`BufReq_ReturnType <Up>_[SoAd][Tp]StartOfReception (`
	`  PduIdType id,`
	`  const PduInfoType* info,`
	`  PduLengthType TpSduLength,`
	`  PduLengthType* bufferSizePtr`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the I-PDU.
	info	Pointer to a PduInfoType structure containing the payload data (without protocol information) and payload length of the first frame or single frame of a transport protocol I-PDU reception, and the MetaData related to this PDU. If neither first/single frame data nor MetaData are available, this parameter is set to NULL_PTR.
	TpSduLength	Total length of the N-SDU to be received.
Parameters (inout)	None	
Parameters (out)	bufferSizePtr	Available receive buffer in the receiving module. This parameter will be used to compute the Block Size (BS) in the transport protocol module.

▽

△

Return value	BufReq_ReturnType	BUFREQ_OK: Connection has been accepted. bufferSizePtr indicates the available receive buffer; reception is continued. If no buffer of the requested size is available, a receive buffer size of 0 shall be indicated by bufferSizePtr. BUFREQ_E_NOT_OK: Connection has been rejected; reception is aborted. bufferSizePtr remains unchanged. BUFREQ_E_OVFL: No buffer of the required length can be provided; reception is aborted. bufferSizePtr remains unchanged.
Description	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.	
Available via	<none>	

⌋

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

[SWS_SoAd_00139] Definition of configurable interface <Up>_[SoAd][Tp]Copy RxData ⌈

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 buffer is written to the position indicated by bufferSizePtr.	
Available via	<none>	

⌋

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

[SWS_SoAd_00180] Definition of configurable interface <Up>_[SoAd][Tp]RxIndication ⌈

Service Name	<Up>_[SoAd][Tp]RxIndication	

Syntax	`void <Up>_[SoAd][Tp]RxIndication (`
	`  PduIdType id,`
	`  Std_ReturnType result`
	`)`	

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the received I-PDU.
	result	`E_OK`: The PDU was received successfully. `E_NOT_OK`: Reception of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Called after an I-PDU has been received via the TP API, the result indicates whether the reception was successful or not.	
Available via	<none>	

⌋

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

[SWS_SoAd_00137] Definition of configurable interface <Up>_[SoAd][Tp]CopyTxData ⌈

Service Name	<Up>_[SoAd][Tp]CopyTxData	

Syntax	`BufReq_ReturnType <Up>_[SoAd][Tp]CopyTxData (`
	`  PduIdType id,`
	`  const PduInfoType* info,`
	`  const 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 Sdu DataPtr 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.	
Available via	<none>	

⌋

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

[SWS_SoAd_00181] Definition of configurable interface <Up>_[SoAd][Tp]TxConfirmation ⌈

Service Name	<Up>_[SoAd][Tp]TxConfirmation	

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

Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	id	Identification of the transmitted I-PDU.
	result	`E_OK`: The PDU was transmitted successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.
Available via	<none>

⌋

8.5.3.9 <Up>_SoConModeChg

[SWS_SoAd_00514] Definition of configurable interface <Up>_SoConModeChg ⌈

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 mode change, e.g. socket connection gets online.	
Available via	<none>	

⌋

8.5.3.10 <Up>_LocalIpAddrAssignmentChg

[SWS_SoAd_00513] Definition of configurable interface <Up>_LocalIpAddrAssignmentChg ⌈

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).
Available via	<none>

⌋

9 Sequence diagrams and Transition Tables

9.1 Address - Assignment

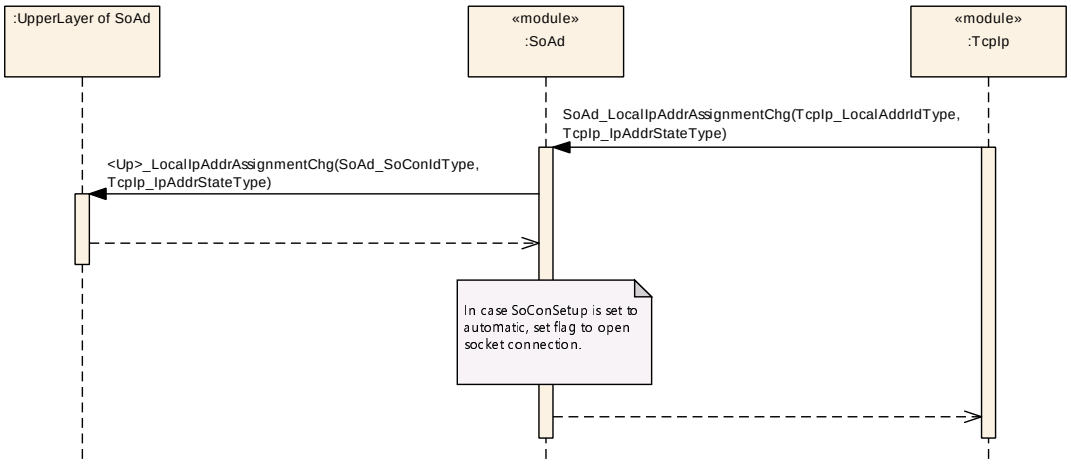


Figure 9.1: SoAd Address Assignment

9.2 Socket Connection Setup - UDP

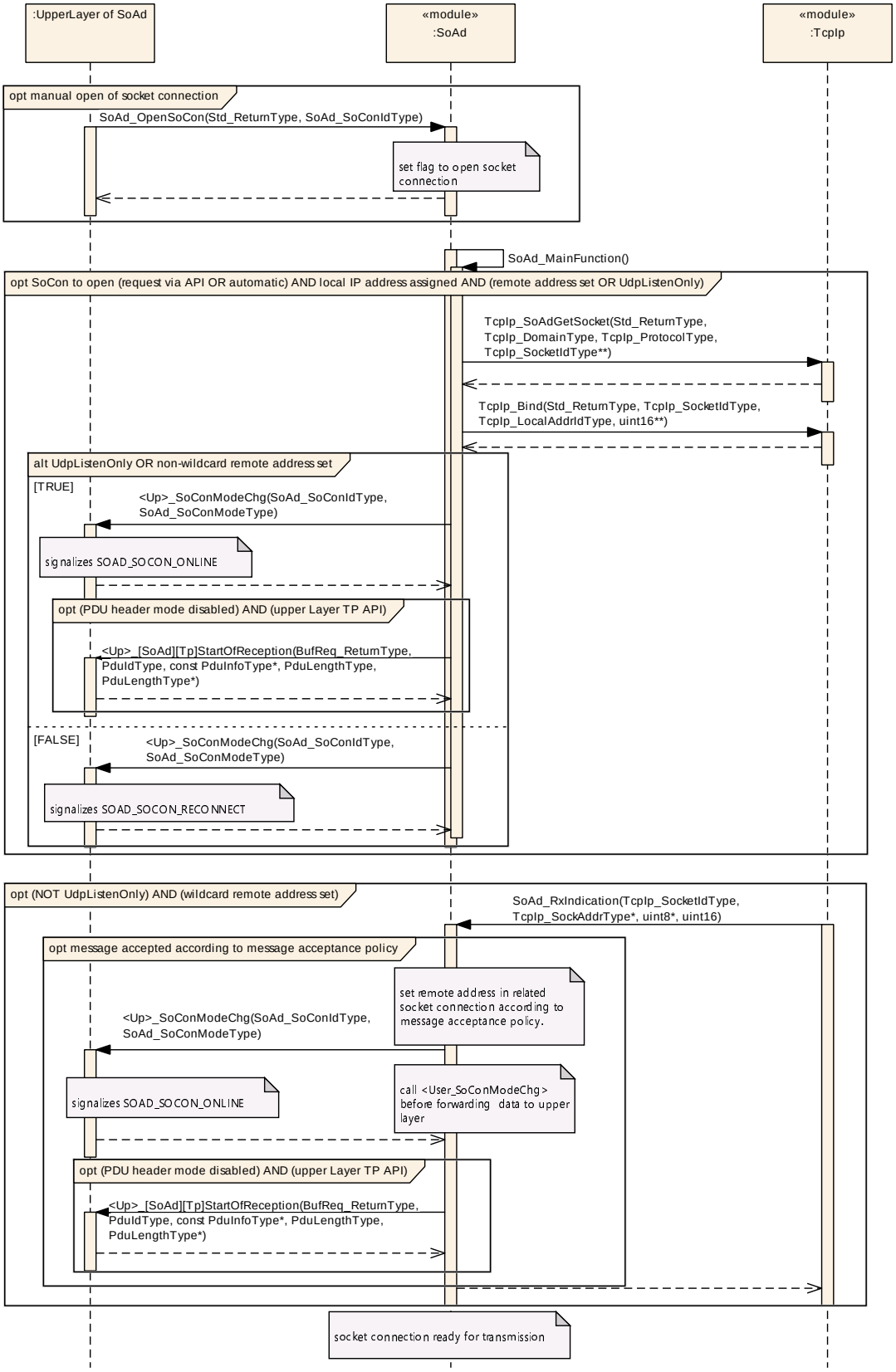


Figure 9.2: SoAd Socket Connection Setup for UDP

9.3 Socket Connection Setup - TCP

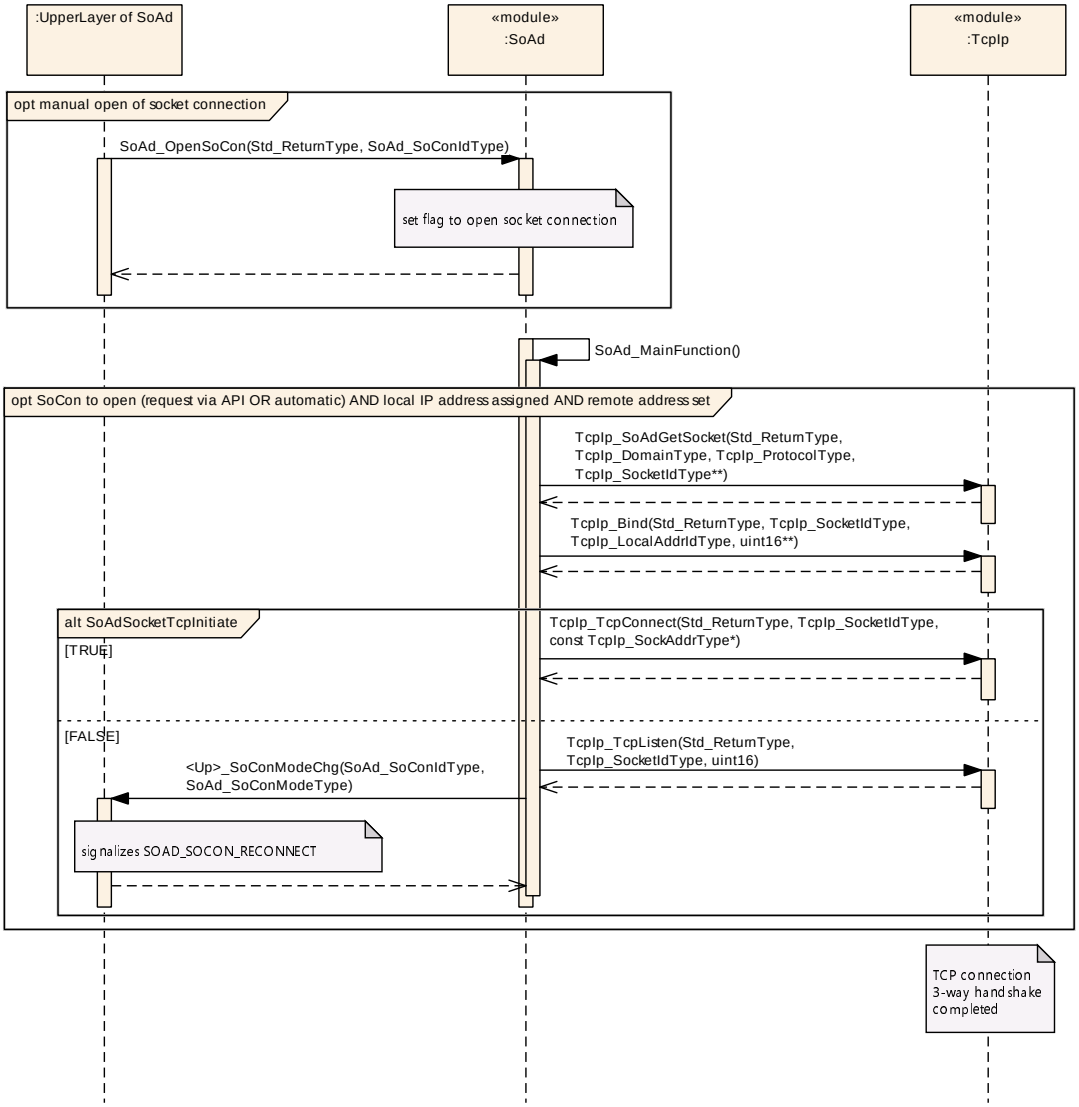


Figure 9.3: SoAd Socket Connection Setup for TCP Part 1

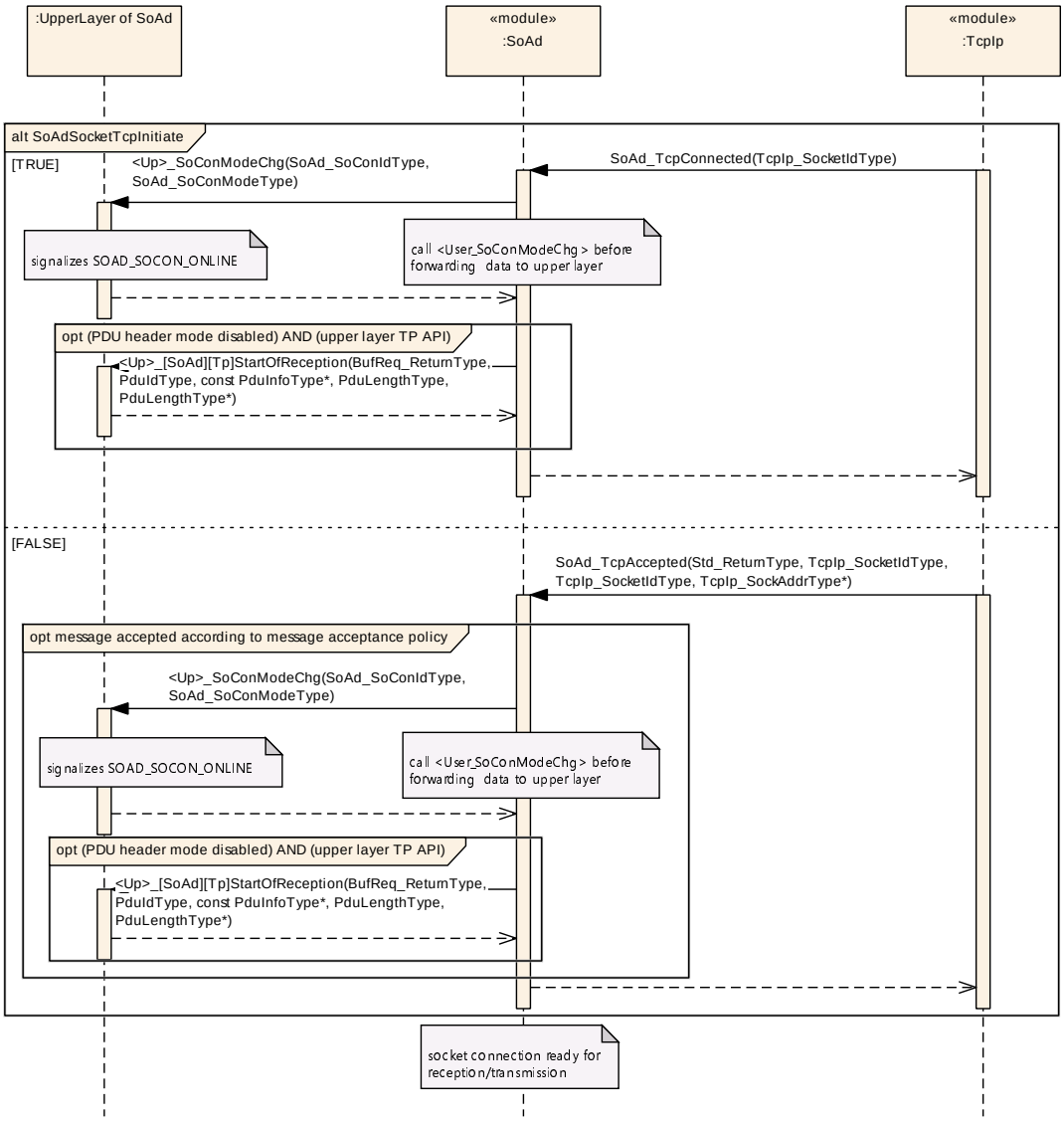


Figure 9.4: SoAd Socket Connection Setup for TCP Part 2

9.4 Reception - Upper Layer If API

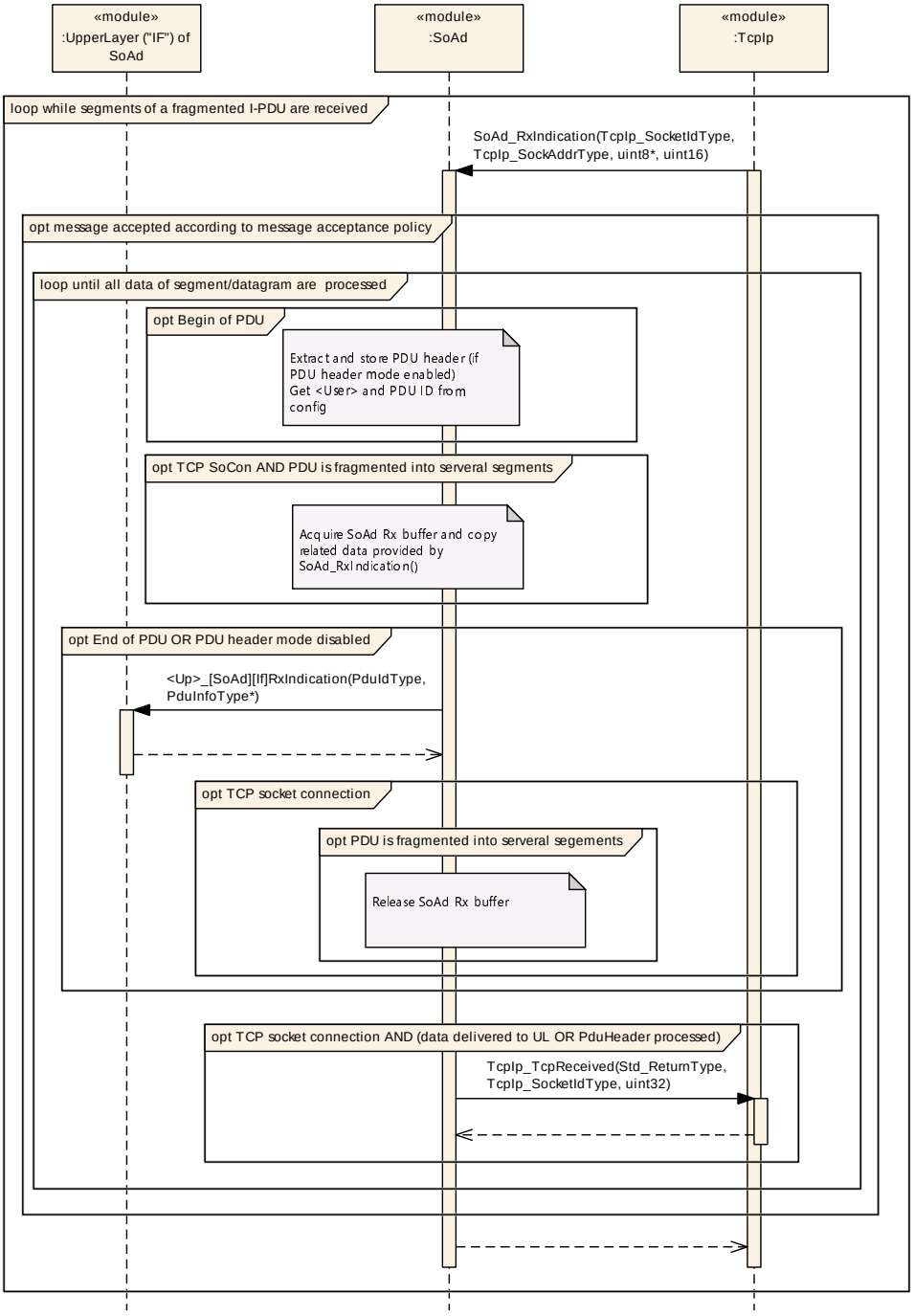


Figure 9.5: Reception for Upper Layer If API

9.5 Reception - Upper Layer TP API

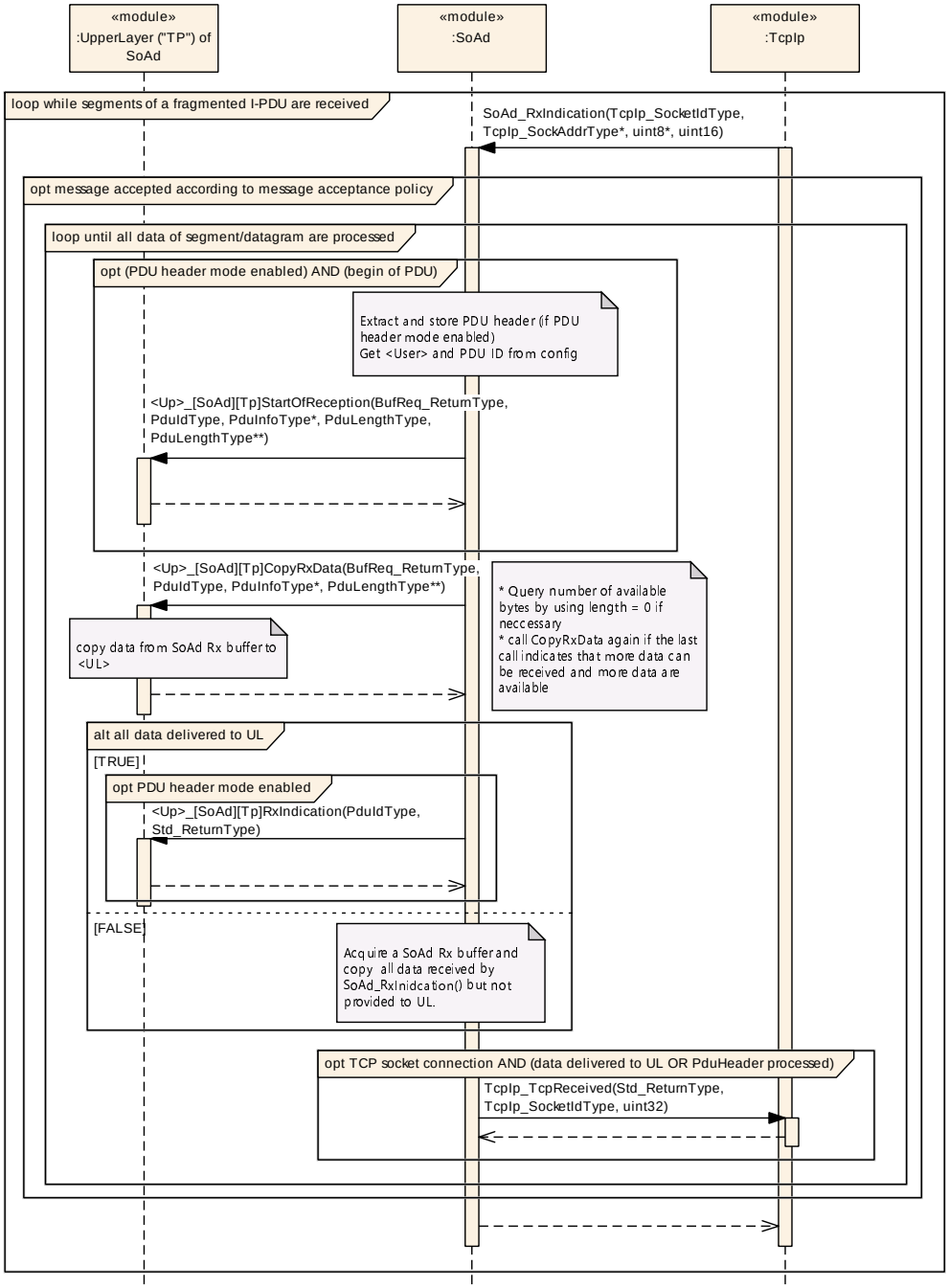


Figure 9.6: Reception for Upper Layer TP API Part 1

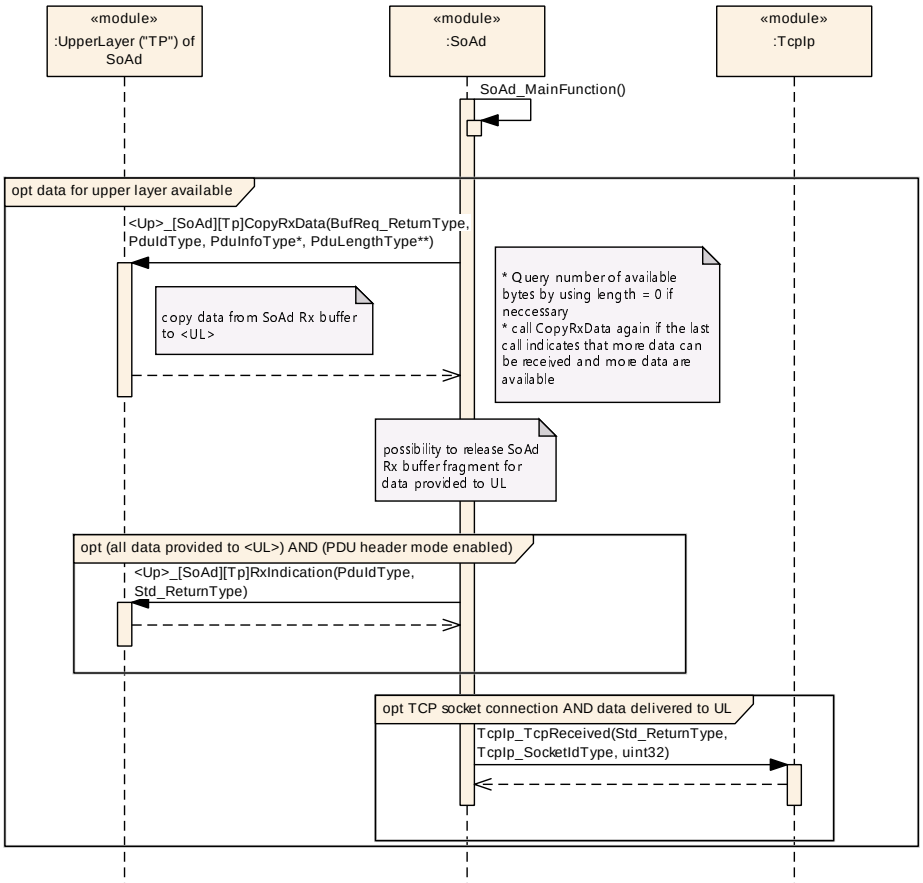


Figure 9.7: Reception for Upper Layer TP API Part 2

9.6 Transmission - Upper Layer If API - TCP

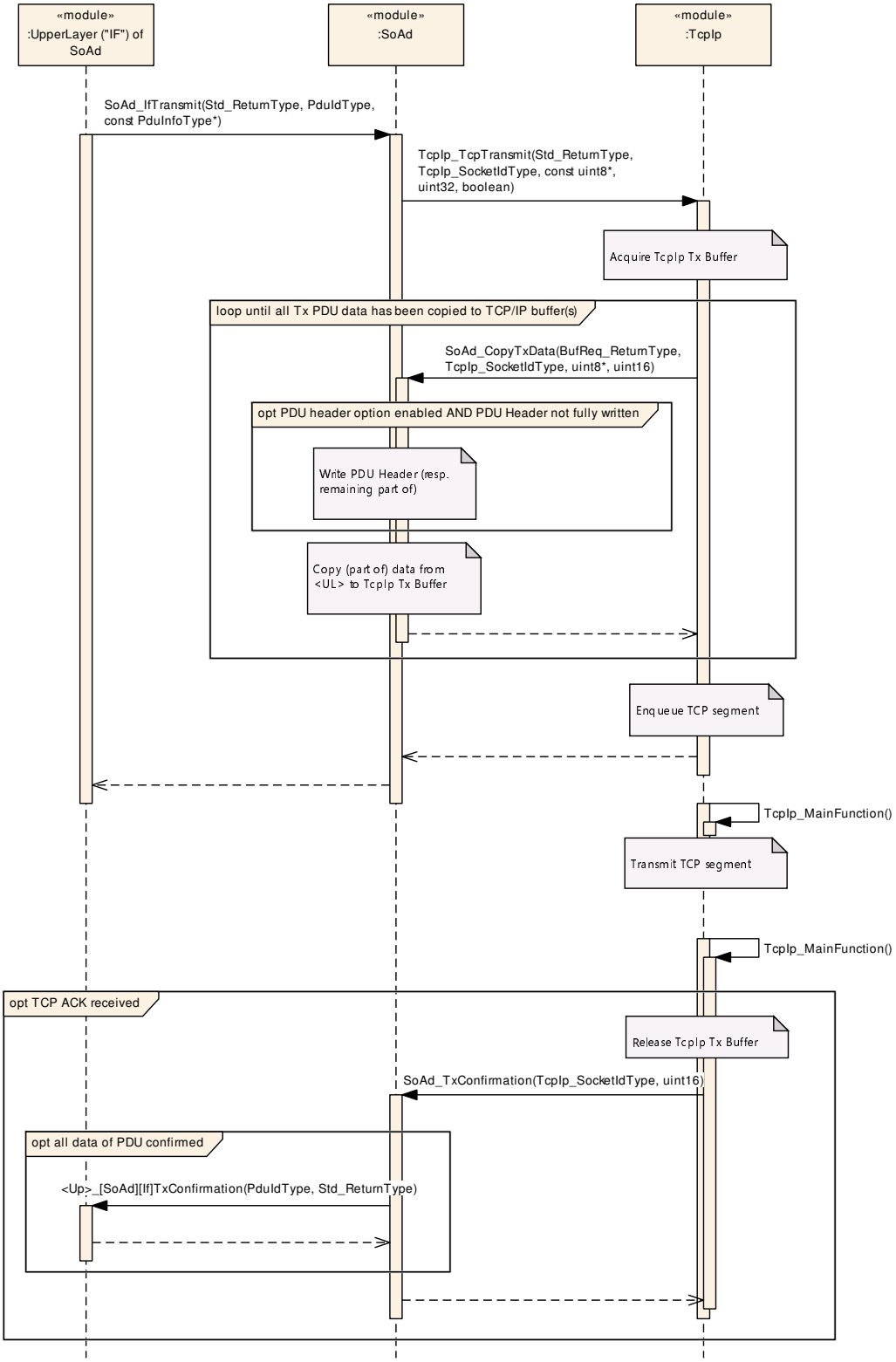


Figure 9.8: Transmission for Upper Layer If API over TCP

9.7 Transmission - Upper Layer If API - UDP

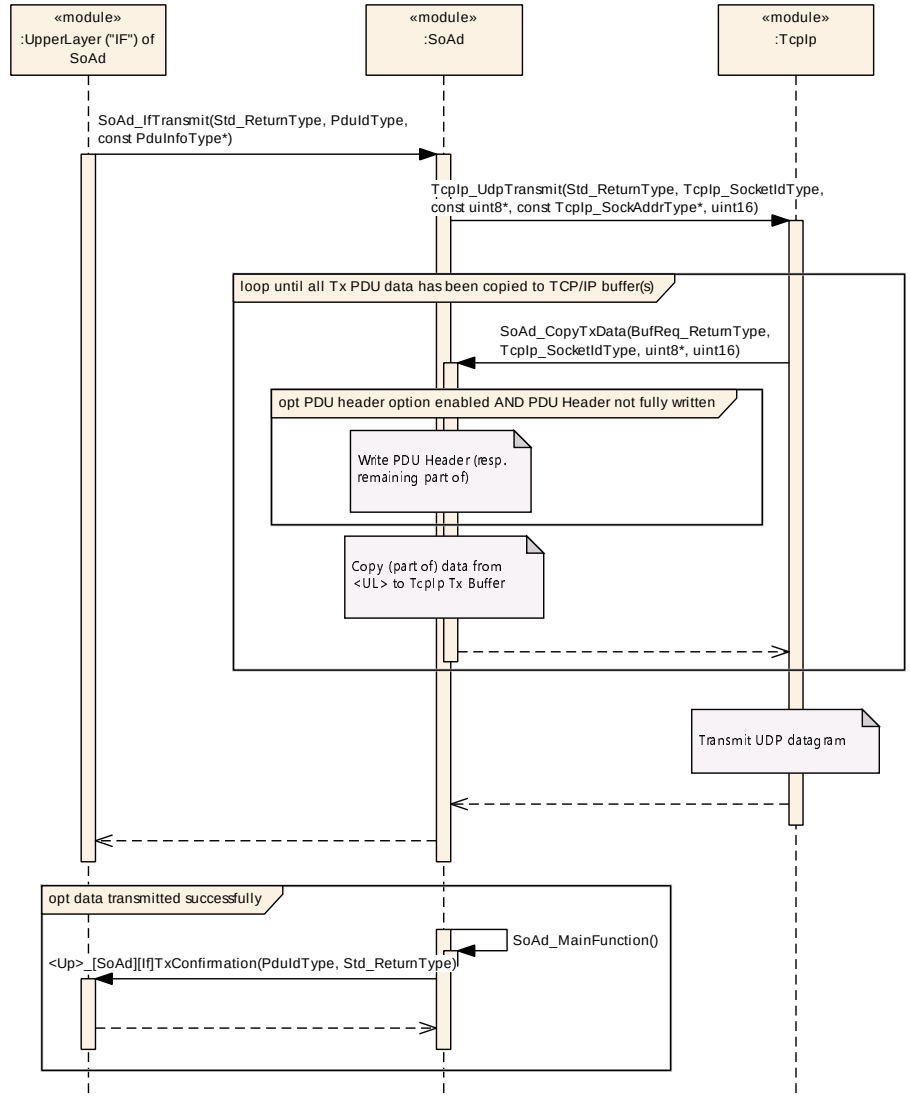


Figure 9.9: Transmission for Upper Layer If API over UDP

9.8 Transmission - Upper Layer TP API

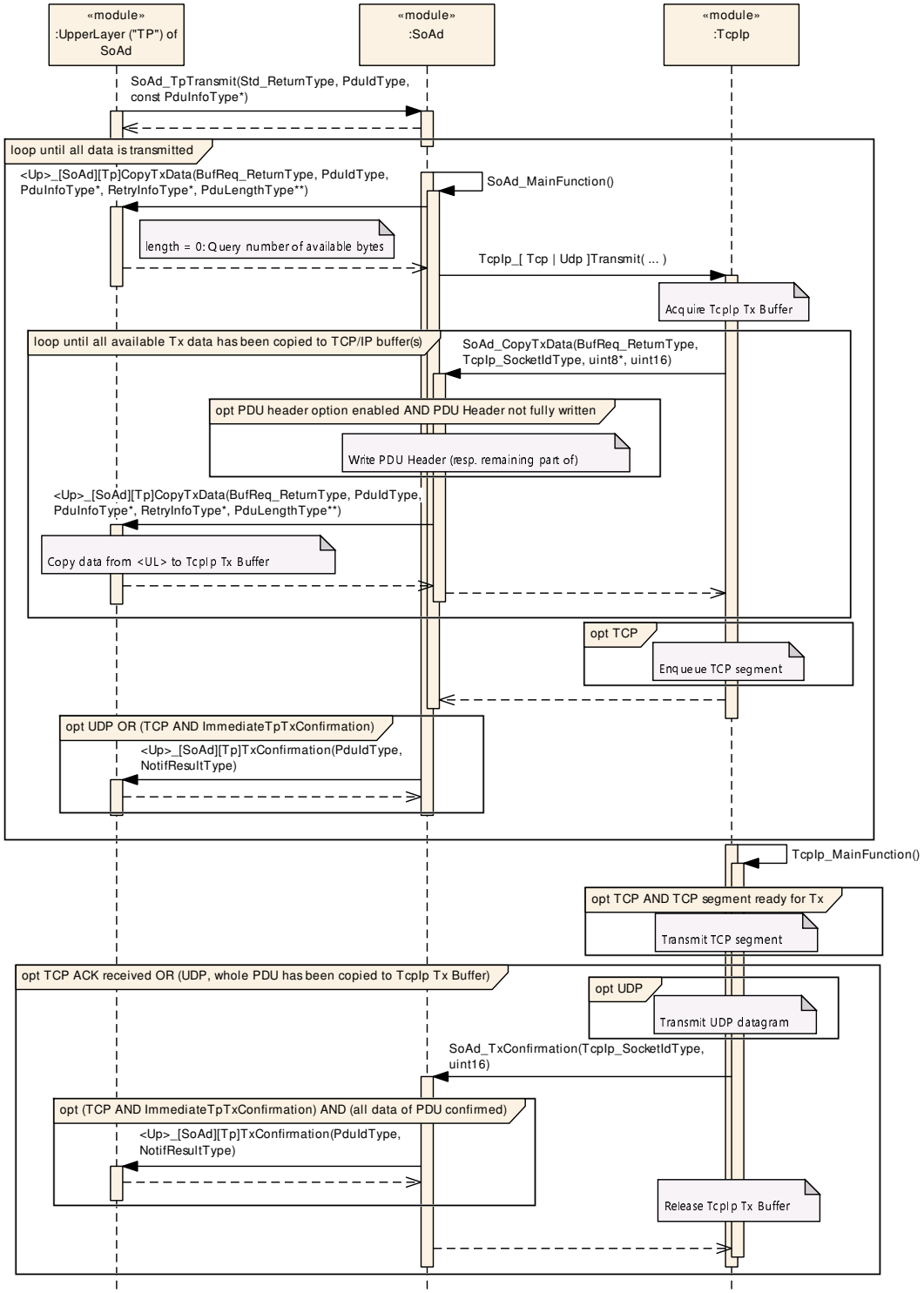


Figure 9.10: Transmission for Upper Layer TP API

10 Configuration specification

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

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

Chapter 10.3 specifies configuration constraints of the module SoAd.

Chapter 10.4 specifies published information of the module SoAd.

10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

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

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

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

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

10.2.1 SoAd

[ECUC_SoAd_00001] Definition of EcucModuleDef SoAd ⌈

Module Name	SoAd
Description	Configuration of the SoAd (Socket Adaptor) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
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 So Ad.

⌋

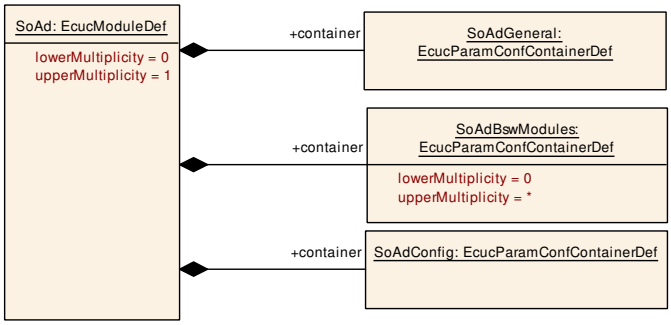


Figure 10.1: SoAd container

10.2.2 SoAdBswModules

[ECUC_SoAd_00102] Definition of EcucParamConfContainerDef SoAdBswModules ⌈

Container Name	SoAdBswModules
Parent Container	SoAd
Upstream mapping defined	–
Description	Each container describes a specific BSW module that the SoAd shall interface to.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdIf	1	[ECUC_SoAd_00104]
SoAdIfTriggerTransmit	1	[ECUC_SoAd_00145]
SoAdIfTxConfirmation	1	[ECUC_SoAd_00106]
SoAdLocalIpAddrAssigmentChg	1	[ECUC_SoAd_00143]
SoAdSoConModeChg	1	[ECUC_SoAd_00107]
SoAdTp	1	[ECUC_SoAd_00105]
SoAdUseCallerInfix	1	[ECUC_SoAd_00128]
SoAdUseTypeInfix	1	[ECUC_SoAd_00129]
SoAdBswModuleRef	1	[ECUC_SoAd_00124]

No Included Containers

⌋

[ECUC_SoAd_00104] Definition of EcucBooleanParamDef SoAdIf ⌈

Parameter Name	SoAdIf
Parent Container	SoAdBswModules
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00145] Definition of EcucBooleanParamDef SoAdIfTriggerTransmit ⌈

Parameter Name	SoAdIfTriggerTransmit
Parent Container	SoAdBswModules
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_SoAd_00106] Definition of EcucBooleanParamDef SoAdIfTxConfirmation ⌈

Parameter Name	SoAdIfTxConfirmation
Parent Container	SoAdBswModules
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00143] Definition of EcucBooleanParamDef SoAdLocalIpAddrAssigmentChg ⌈

Parameter Name	SoAdLocalIpAddrAssigmentChg
Parent Container	SoAdBswModules
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00107] Definition of EcucBooleanParamDef SoAdSoConModeChg ⌈

Parameter Name	SoAdSoConModeChg
Parent Container	SoAdBswModules
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00105] Definition of EcucBooleanParamDef SoAdTp ⌈

Parameter Name	SoAdTp
Parent Container	SoAdBswModules
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00128] Definition of EcucBooleanParamDef SoAdUseCallerInfix ⌈

Parameter Name	SoAdUseCallerInfix
Parent Container	SoAdBswModules
Upstream mapping defined	–

▽

△

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_IfRx Indication().
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_SoAd_00129] Definition of EcucBooleanParamDef SoAdUseTypeInfix ⌈

Parameter Name	SoAdUseTypeInfix
Parent Container	SoAdBswModules
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_SoAd_00124] Definition of EcucForeignReferenceDef SoAdBswModule Ref ⌈

Parameter Name	SoAdBswModuleRef
Parent Container	SoAdBswModules
Upstream mapping defined	[11, CP_TPS_ECUConfiguration]
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
Pre-Compile only	true

⌋

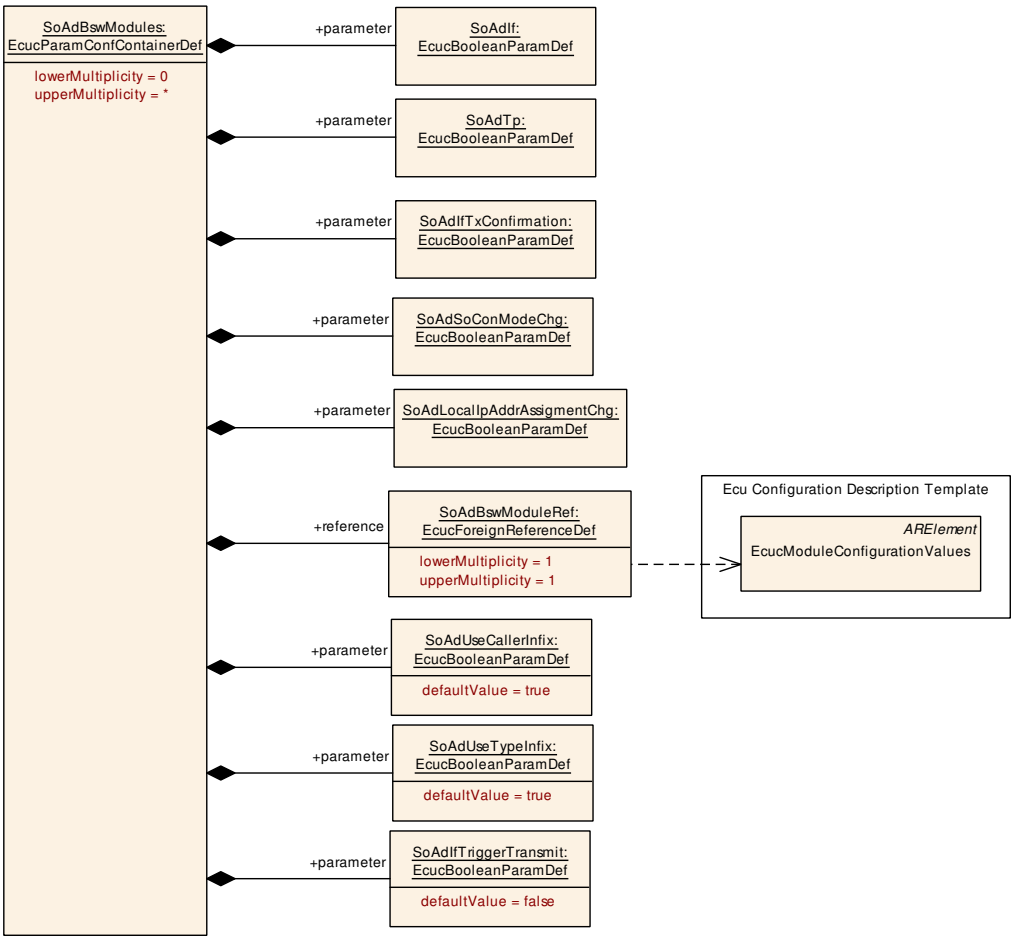


Figure 10.2: SoAd BswModules container

10.2.3 SoAdGeneral

[ECUC_SoAd_00003] Definition of EcucParamConfContainerDef SoAdGeneral ⌈

Container Name	SoAdGeneral
Parent Container	SoAd
Upstream mapping defined	–
Description	This container contains all global configuration parameters of SoAd.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdDevErrorDetect	1	[ECUC_SoAd_00002]
SoAdEnableSecurityEventReporting	1	[ECUC_SoAd_00164]
SoAdGetAndResetMeasurementDataApi	1	[ECUC_SoAd_00162]

⌋

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdIPv6AddressEnabled	1	[ECUC_SoAd_00039]
SoAdMainFunctionPeriod	1	[ECUC_SoAd_00062]
SoAdRoutingGroupMax	1	[ECUC_SoAd_00127]
SoAdSoConMax	1	[ECUC_SoAd_00126]

Included Containers		
Container Name	**Multiplicity**	**Description**
SoAdSecurityEventRefs	0..1	Container for the references to IdsMEvent elements representing the security events that the SoAd module shall report to the IdsM in case the coresponding security related event occurs (and if SoAdEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft

⌋

[ECUC_SoAd_00002] Definition of EcucBooleanParamDef SoAdDevErrorDetect

⌈

Parameter Name	SoAdDevErrorDetect
Parent Container	SoAdGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_SoAd_00164] Definition of EcucBooleanParamDef SoAdEnableSecurityEventReporting

Status: DRAFT

⌈

Parameter Name	SoAdEnableSecurityEventReporting
Parent Container	SoAdGeneral
Upstream mapping defined	–
Description	Switches the reporting of security events to the IdsM: - true: reporting is enabled. - false: reporting is disabled. **Tags:** atp.Status=draft
Multiplicity	1
Type	EcucBooleanParamDef

▽

△

Default value	false
Pre-Compile only	true

⌋

[ECUC_SoAd_00162] Definition of EcucBooleanParamDef SoAdGetAndReset MeasurementDataApi ⌈

Parameter Name	SoAdGetAndResetMeasurementDataApi
Parent Container	SoAdGeneral
Upstream mapping defined	–
Description	Enables / Disables the Get and Reset Measurement Data API
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00039] Definition of EcucBooleanParamDef SoAdIPv6AddressEnabled ⌈

Parameter Name	SoAdIPv6AddressEnabled
Parent Container	SoAdGeneral
Upstream mapping defined	–
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	–
Pre-Compile only	true

⌋

[ECUC_SoAd_00062] Definition of EcucFloatParamDef SoAdMainFunctionPeriod ⌈

Parameter Name	SoAdMainFunctionPeriod	
Parent Container	SoAdGeneral	
Upstream mapping defined	–	
Description	Determines the frequency at which the SoAd_MainFunction() is called in [s].	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SoAd_00127] Definition of EcucIntegerParamDef SoAdRoutingGroupMax ⌈

Parameter Name	SoAdRoutingGroupMax	
Parent Container	SoAdGeneral	
Upstream mapping defined	–	
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, a uint8 is used, otherwise a uint16.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_SoAd_00126] Definition of EcucIntegerParamDef SoAdSoConMax ⌈

Parameter Name	SoAdSoConMax	
Parent Container	SoAdGeneral	
Upstream mapping defined	–	
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, a uint8 is used, otherwise uint16.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Pre-Compile only	true	

⌋

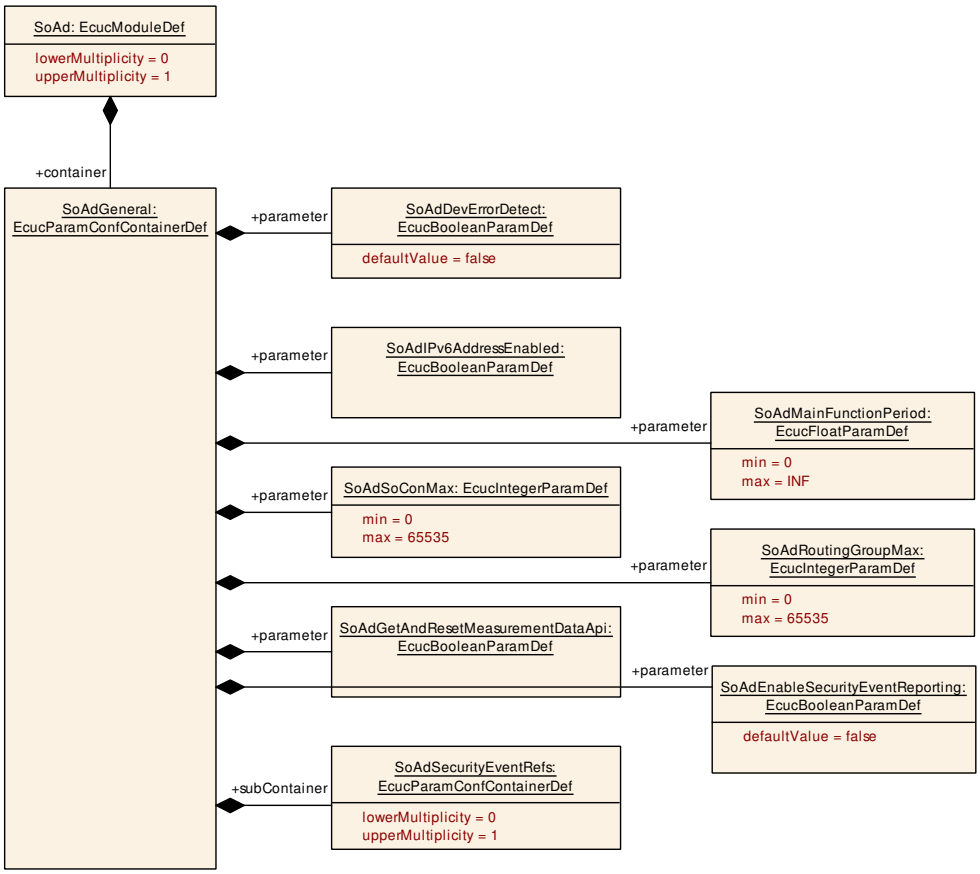


Figure 10.3: SoAd General container

10.2.4 SoAdSecurityEventRefs

[ECUC_SoAd_00165] Definition of EcucParamConfContainerDef SoAdSecurity EventRefs

Status: DRAFT

⌈

Container Name	SoAdSecurityEventRefs
Parent Container	SoAdGeneral
Upstream mapping defined	–
Description	Container for the references to IdsMEvent elements representing the security events that the SoAd module shall report to the IdsM in case the coresponding security related event occurs (and if SoAdEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events. **Tags:** atp.Status=draft
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SEV_DROP_MSG_RX_UDP_LENGTH	0..1	[ECUC_SoAd_00168]
SEV_DROP_MSG_RX_UDP_SOCKET	0..1	[ECUC_SoAd_00169]
SEV_DROP_PDU_RX_TCP	0..1	[ECUC_SoAd_00166]
SEV_DROP_PDU_RX_UDP	0..1	[ECUC_SoAd_00167]
SEV_REJECTED_TCP_CONNECTION	0..1	[ECUC_SoAd_00170]

No Included Containers

⌋

[ECUC_SoAd_00168] Definition of EcucReferenceDef SEV_DROP_MSG_RX_UDP_LENGTH

Status: DRAFT

⌈

Parameter Name	SEV_DROP_MSG_RX_UDP_LENGTH
Parent Container	SoAdSecurityEventRefs
Upstream mapping defined	–
Description	SoAd dropped a message. The message contains at least one PDU which violates stack configuration and was received via a UDP socket. The violation relates to the length of the PDUs compared to the overall length of the message. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_SoAd_00169] Definition of EcucReferenceDef SEV_DROP_MSG_RX_UDP_SOCKET

Status: DRAFT

⌈

Parameter Name	SEV_DROP_MSG_RX_UDP_SOCKET
Parent Container	SoAdSecurityEventRefs
Upstream mapping defined	–
Description	SoAd received a UDP message which violates stack configuration and was dropped. No suitable socket connection matching to configuration was found. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_SoAd_00166] Definition of EcucReferenceDef SEV_DROP_PDU_RX_TCP

Status: DRAFT

⌈

Parameter Name	SEV_DROP_PDU_RX_TCP
Parent Container	SoAdSecurityEventRefs
Upstream mapping defined	–
Description	SoAd dropped a PDU. The PDU violates stack configuration and was received via a TCP socket. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_SoAd_00167] Definition of EcucReferenceDef SEV_DROP_PDU_RX_UDP

Status: DRAFT

⌈

Parameter Name	SEV_DROP_PDU_RX_UDP
Parent Container	SoAdSecurityEventRefs
Upstream mapping defined	–
Description	SoAd dropped a PDU. The PDU violates stack configuration and was received via a UDP socket. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

[ECUC_SoAd_00170] Definition of EcucReferenceDef SEV_REJECTED_TCP_CONNECTION

Status: DRAFT

⌈

Parameter Name	SEV_REJECTED_TCP_CONNECTION
Parent Container	SoAdSecurityEventRefs
Upstream mapping defined	–
Description	SoAd rejected a TCP connection. The connection request violates stack configuration. **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Symbolic name reference to IdsMEvent
Pre-Compile only	true

⌋

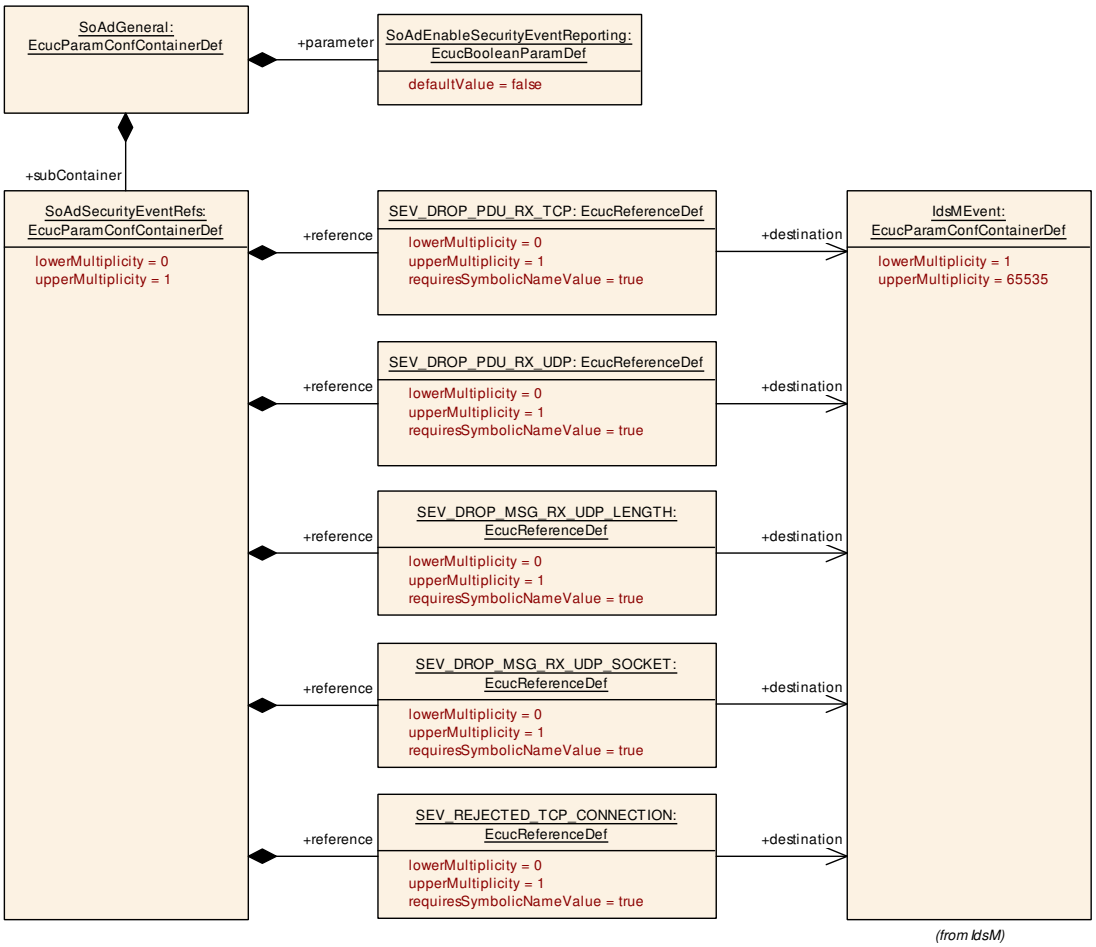


Figure 10.4: SoAd Security Events DRAFT

10.2.5 SoAdConfig

[ECUC_SoAd_00103] Definition of EcucParamConfContainerDef SoAdConfig ⌈

Container Name	SoAdConfig
Parent Container	SoAd
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR SoAd module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
SoAdPduRoute	0..*	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission. This PDU can consume meta data items of type SOCKET_CONNECTION_ID_16.
SoAdRoutingGroup	0..*	Each container describes a specific routing group which can be enabled or disabled. A routing group consists of PDUs. Routing of PDUs can either be forwarding of PDUs from the upper layer to a TCP or UDP socket of the TCP/IP stack specified by a SoAd PduRoute or the other way around specified by a SoAdSocket Route.
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.

⌋

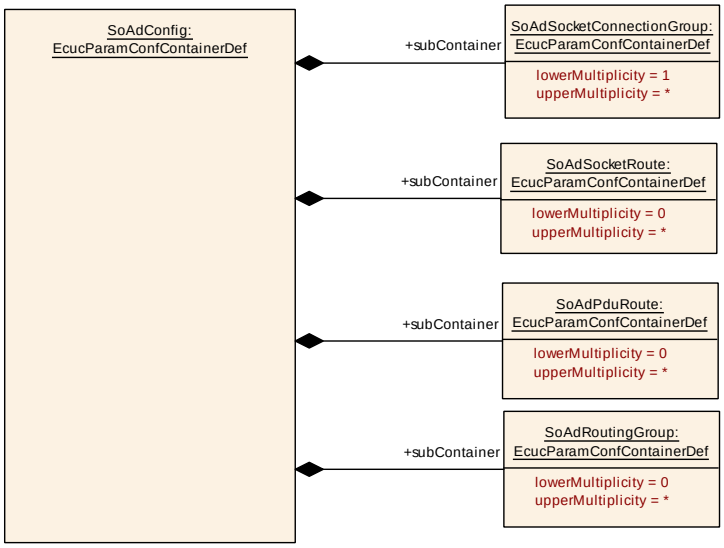


Figure 10.5: SoAd Config container

10.2.6 SoAdSocketConnectionGroup

[ECUC_SoAd_00130] Definition of EcucParamConfContainerDef SoAdSocket ConnectionGroup ⌈

Container Name	SoAdSocketConnectionGroup
Parent Container	SoAdConfig
Upstream mapping defined	[12, CP_TPS_SystemTemplate]

▽

△

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.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdPduHeaderEnable	1	[ECUC_SoAd_00131]
SoAdSocketAutomaticSoConSetup	1	[ECUC_SoAd_00110]
SoAdSocketDifferentiatedServicesField	0..1	[ECUC_SoAd_00158]
SoAdSocketFlowLabel	0..1	[ECUC_SoAd_00157]
SoAdSocketFramePriority	0..1	[ECUC_SoAd_00138]
SoAdSocketIpAddrAssignmentChgNotification	1	[ECUC_SoAd_00112]
SoAdSocketLocalPort	1	[ECUC_SoAd_00018]
SoAdSocketMsgAcceptanceFilterEnabled	1	[ECUC_SoAd_00137]
SoAdSocketSoConModeChgBswMNotification	0..1	[ECUC_SoAd_00173]
SoAdSocketSoConModeChgNotification	1	[ECUC_SoAd_00111]
SoAdSocketTimeToLive	0..1	[ECUC_SoAd_00178]
SoAdSocketTpRxBufferMin	0..1	[ECUC_SoAd_00134]
SoAdSocketIpAddrAssignmentChgNotifUpperLayerRef	0..1	[ECUC_SoAd_00175]
SoAdSocketLocalAddressRef	1	[ECUC_SoAd_00017]
SoAdSocketSoConModeChgNotifUpperLayerRef	0..1	[ECUC_SoAd_00161]

Included Containers		
Container Name	**Multiplicity**	**Description**
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

⌋

[ECUC_SoAd_00131] Definition of EcucBooleanParamDef SoAdPduHeaderEnable ⌈

Parameter Name	SoAdPduHeaderEnable	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00110] Definition of EcucBooleanParamDef SoAdSocketAutomaticSoConSetup ⌈

Parameter Name	SoAdSocketAutomaticSoConSetup	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	–	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00158] Definition of EcucIntegerParamDef SoAdSocketDifferentiatedServicesField ⌈

Parameter Name	SoAdSocketDifferentiatedServicesField	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	The 6-bit Differentiated Service Field in the IP headers may be used for classifying network traffic. If not set a value of zero is used to indicate packets that have not been classified.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 63	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00157] Definition of EcucIntegerParamDef SoAdSocketFlowLabel ⌈

Parameter Name	SoAdSocketFlowLabel	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	The 20-bit Flow Label field in the IPv6 header may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service. If not set a Flow Label of zero is used to indicate packets that have not been labeled.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 1048575	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00138] Definition of EcucIntegerParamDef SoAdSocketFramePriority ⌈

Parameter Name	SoAdSocketFramePriority	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00112] Definition of EcucBooleanParamDef SoAdSocketIpAddrAssignmentChgNotification ⌈

Parameter Name	SoAdSocketIpAddrAssignmentChgNotification	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	–	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00018] Definition of EcucIntegerParamDef SoAdSocketLocalPort ⌈

Parameter Name	SoAdSocketLocalPort	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00137] Definition of EcucBooleanParamDef SoAdSocketMsgAcceptanceFilterEnabled ⌈

Parameter Name	SoAdSocketMsgAcceptanceFilterEnabled	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	–	
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. Dependency: SoAdSocketRemoteAddress, SoAdSocketUdpListenOnly, SoAdSocket ConnectionGroup	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	true	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00173] Definition of EcucBooleanParamDef SoAdSocketSoCon ModeChgBswMNotification ⌈

Parameter Name	SoAdSocketSoConModeChgBswMNotification	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	–	
Description	Specifies if the BswM_SoAd_SoConModeChg notification shall be called in case of So Con mode change	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00111] Definition of EcucBooleanParamDef SoAdSocketSoCon ModeChgNotification ⌈

Parameter Name	SoAdSocketSoConModeChgNotification	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	–	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00178] Definition of EcucIntegerParamDef SoAdSocketTimeToLive ⌈

Parameter Name	SoAdSocketTimeToLive	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	This parameter defines a value set in the header of an Internet Protocol (IP) packet that tells network devices the maximum number of router hops the packet can make before it is discarded. The TTL value is a counter that is decremented by 1 every time the packet passes through a router.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 255	
Default value	–	
Post-Build Multiplicity	-	Selectable
Post-Build Value	-	Selectable

⌋

[ECUC_SoAd_00134] Definition of EcucIntegerParamDef SoAdSocketTpRxBufferMin ⌈

Parameter Name	SoAdSocketTpRxBufferMin	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00175] Definition of EcucReferenceDef SoAdSocketIpAddrAssignmentChgNotifUpperLayerRef

Status: DRAFT

⌈

Parameter Name	SoAdSocketIpAddrAssignmentChgNotifUpperLayerRef
Parent Container	SoAdSocketConnectionGroup
Upstream mapping defined	–

▽

△

Description	Reference to an additional upper layer that shall be called if the assignment of the local IP address used by this socket connection changes (although it is not a direct upper layer of the socket connection). **Tags:** atp.Status=draft
Multiplicity	0..1
Type	Reference to SoAdBswModules
Pre-Compile only	true

⌋

[ECUC_SoAd_00017] Definition of EcucReferenceDef SoAdSocketLocalAddressRef ⌈

Parameter Name	SoAdSocketLocalAddressRef	
Parent Container	SoAdSocketConnectionGroup	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Local IP address and interface used for this connection.	
Multiplicity	1	
Type	Symbolic name reference to TcpIpLocalAddr	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00161] Definition of EcucReferenceDef SoAdSocketSoConModeChgNotifUpperLayerRef ⌈

Parameter Name	SoAdSocketSoConModeChgNotifUpperLayerRef
Parent Container	SoAdSocketConnectionGroup
Upstream mapping defined	–
Description	Reference to an additional upper layer that shall receive socket connection state changes (although it is not a direct upper layer of the socket connection).
Multiplicity	0..1
Type	Reference to SoAdBswModules
Pre-Compile only	true

⌋

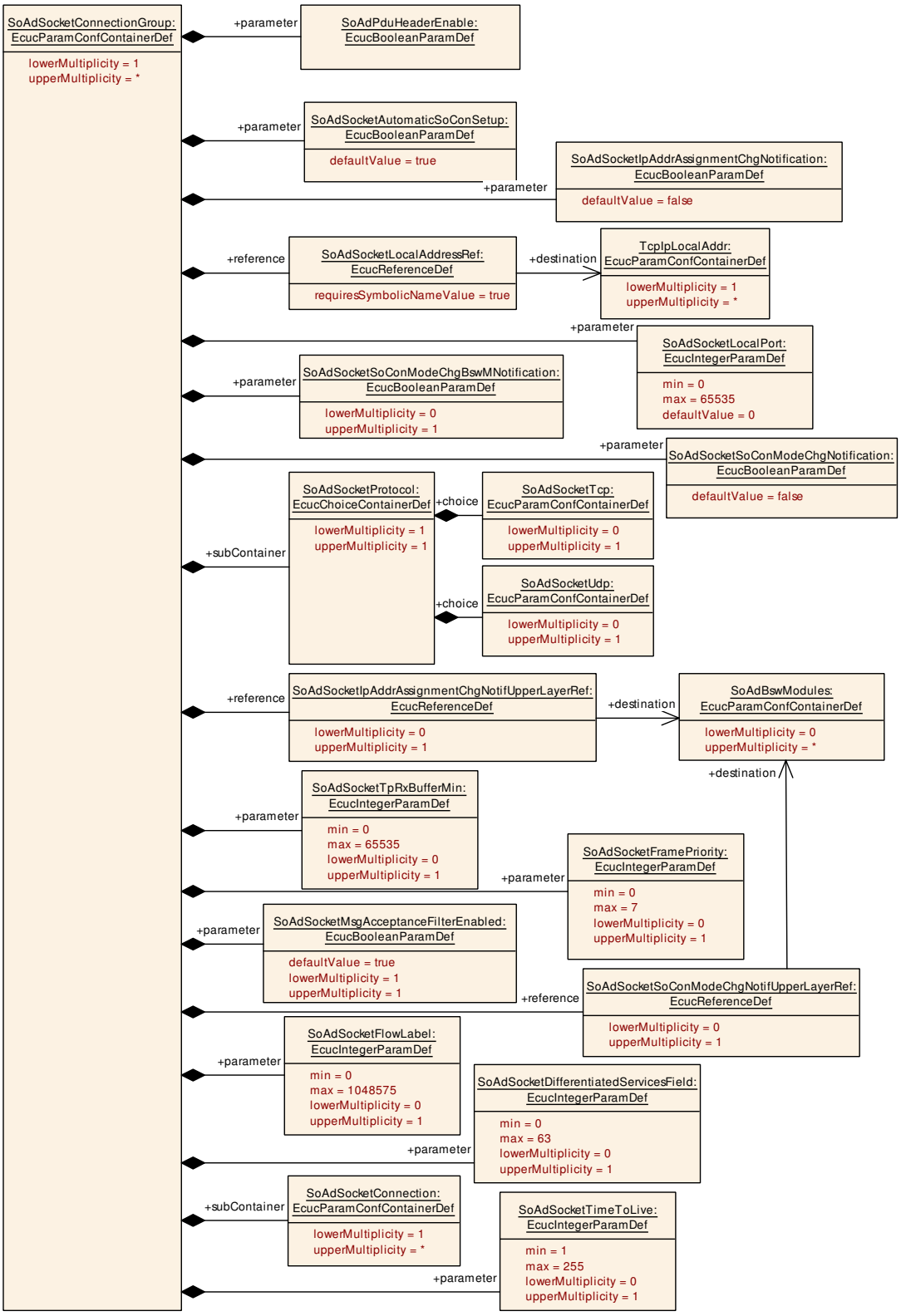


Figure 10.6: SoAd Socket Connection Group container

10.2.7 SoAdSocketConnection

[ECUC_SoAd_00009] Definition of EcucParamConfContainerDef SoAdSocketConnection ⌈

Container Name	SoAdSocketConnection
Parent Container	SoAdSocketConnectionGroup
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
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.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdSocketId	1	[ECUC_SoAd_00016]

Included Containers		
Container Name	**Multiplicity**	**Description**
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().

⌋

[ECUC_SoAd_00016] Definition of EcucIntegerParamDef SoAdSocketId ⌈

Parameter Name	SoAdSocketId	
Parent Container	SoAdSocketConnection	
Upstream mapping defined	–	
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

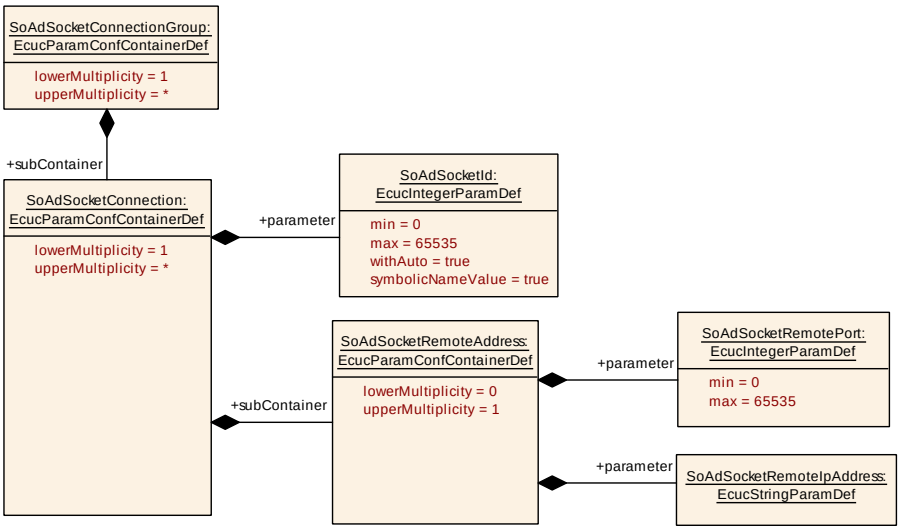


Figure 10.7: SoAd Socket Connection container

10.2.8 SoAdSocketProtocol

[ECUC_SoAd_00139] Definition of EcucChoiceContainerDef SoAdSocketProtocol ⌈

Choice Container Name	SoAdSocketProtocol
Parent Container	SoAdSocketConnectionGroup
Upstream mapping defined	–
Description	Specifies the transport protocol and transport protocol specific parameters used for the socket connections of the socket connection group.
Multiplicity	1
Pre-Compile only	true

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Description**
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

[ECUC_SoAd_00140] Definition of EcucParamConfContainerDef SoAdSocketUdp ⌈

Container Name	SoAdSocketUdp
Parent Container	SoAdSocketProtocol
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
Description	Specifies that UDP is used as transport protocol for the socket connection group and parameters only related to UDP socket connections.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SoAdSocketnPduUdpTxBufferMin	0..1	[ECUC_SoAd_00135]
SoAdSocketUdpAliveSupervisionTimeout	0..1	[ECUC_SoAd_00149]
SoAdSocketUdpChecksumEnabled	1	[ECUC_SoAd_00159]
SoAdSocketUdpListenOnly	1	[ECUC_SoAd_00024]
SoAdSocketUdpStrictHeaderLenCheckEnabled	1	[ECUC_SoAd_00154]
SoAdSocketUdpTriggerTimeout	0..1	[ECUC_SoAd_00133]
SoAdDatagramTlsConnectionRef	0..1	[ECUC_SoAd_00181]

No Included Containers

⌋

[ECUC_SoAd_00135] Definition of EcucIntegerParamDef SoAdSocketnPduUdpTxBufferMin ⌈

Parameter Name	SoAdSocketnPduUdpTxBufferMin	
Parent Container	SoAdSocketUdp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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. Dependency: SoAdTxUdpTriggerMode	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00149] Definition of EcucFloatParamDef SoAdSocketUdpAliveSupervisionTimeout ⌈

Parameter Name	SoAdSocketUdpAliveSupervisionTimeout	
Parent Container	SoAdSocketUdp	
Upstream mapping defined	–	
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 Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00159] Definition of EcucBooleanParamDef SoAdSocketUdpChecksumEnabled ⌈

Parameter Name	SoAdSocketUdpChecksumEnabled	
Parent Container	SoAdSocketUdp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Specifies if UDP checksum calculation shall be enabled (TRUE) or skipped (FALSE) on the related socket. FALSE implies that the upper layer of the socket connection is either capable to handle malformed messages or applies a checksum mechanism itself.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	true	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00024] Definition of EcucBooleanParamDef SoAdSocketUdpListenOnly ⌈

Parameter Name	SoAdSocketUdpListenOnly	
Parent Container	SoAdSocketUdp	
Upstream mapping defined	–	
Description	Specifies if the socket connection group is only used for reception (TRUE) or used for both reception and transmission (FALSE). Dependency: SoAdSocketProtocol	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00154] Definition of EcucBooleanParamDef SoAdSocketUdpStrictHeaderLenCheckEnabled ⌈

Parameter Name	SoAdSocketUdpStrictHeaderLenCheckEnabled	
Parent Container	SoAdSocketUdp	
Upstream mapping defined	–	
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. Dependency: SoAdPduHeaderEnable	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00133] Definition of EcucFloatParamDef SoAdSocketUdpTriggerTimeout ⌈

Parameter Name	SoAdSocketUdpTriggerTimeout	
Parent Container	SoAdSocketUdp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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. Dependency: SoAdTxUdpTriggerMode	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00181] Definition of EcucReferenceDef SoAdDatagramTlsConnectionRef ⌈

Parameter Name	SoAdDatagramTlsConnectionRef
Parent Container	SoAdSocketUdp
Upstream mapping defined	–
Description	Reference to a TcpIpTlsConnection used for DTLS over UDP. Dependency: For a UDP connection with SoAdDatagramTlsConnectionRef set and referencing a TcpIpTlsConnection with DTLS disabled (i.e. TcpIpTlsProtocol does not select a Dtls branch), an error shall be reported.
Multiplicity	0..1
Type	Symbolic name reference to TcpIpTlsConnection
Pre-Compile only	true

⌋

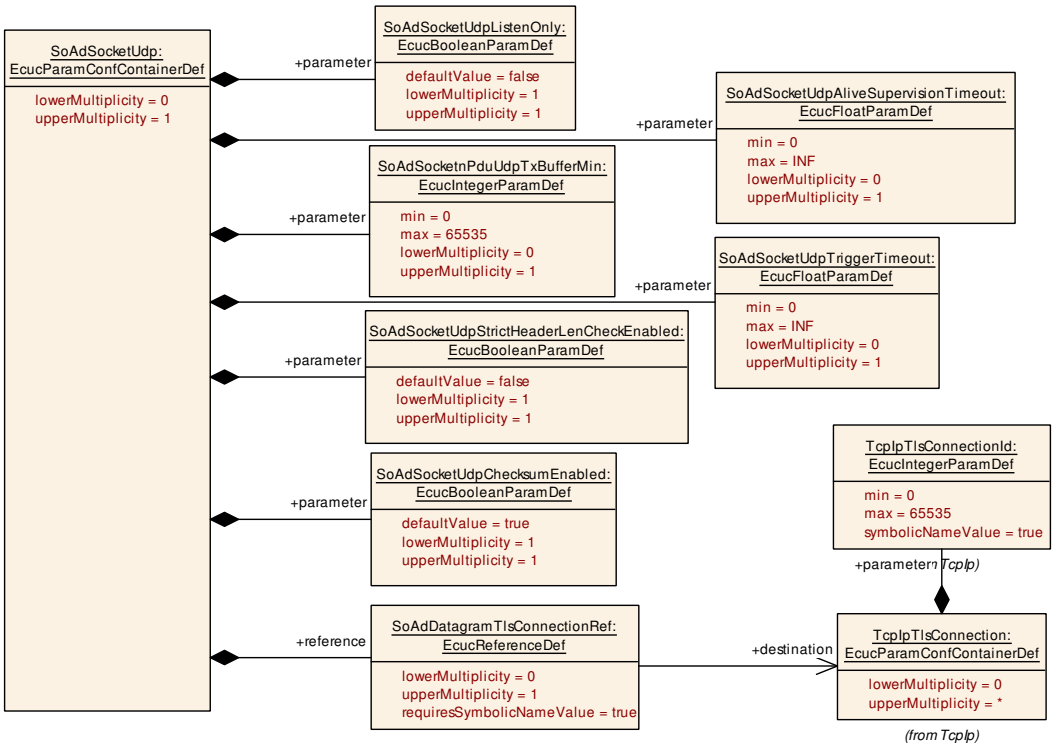


Figure 10.8: SoAd Socket Connection Protocol UDP container

10.2.10 SoAdSocketTcp

[ECUC_SoAd_00141] Definition of EcucParamConfContainerDef SoAdSocketTcp ⌈

Container Name	SoAdSocketTcp
Parent Container	SoAdSocketProtocol
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
Description	Specifies that TCP is used as transport protocol for the socket connection group and parameters only related to TCP socket connections.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdSocketIfImmediateTxConfirmation	1	[ECUC_SoAd_00183]
SoAdSocketTcpAutoConnectTimeout	0..1	[ECUC_SoAd_00174]
SoAdSocketTcpImmediateTpTxConfirmation	1	[ECUC_SoAd_00147]
SoAdSocketTcpInitiate	1	[ECUC_SoAd_00022]
SoAdSocketTcpKeepAlive	1	[ECUC_SoAd_00148]
SoAdSocketTcpKeepAliveInterval	0..1	[ECUC_SoAd_00152]

⌋

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdSocketTcpKeepAliveProbesMax	0..1	[ECUC_SoAd_00151]
SoAdSocketTcpKeepAliveTime	0..1	[ECUC_SoAd_00153]
SoAdSocketTcpMaxRetransmissions	0..1	[ECUC_SoAd_00179]
SoAdSocketTcpMaxRetransmissionTimeout	0..1	[ECUC_SoAd_00180]
SoAdSocketTcpNoDelay	0..1	[ECUC_SoAd_00023]
SoAdSocketTcpRetransmissionTimeout	0..1	[ECUC_SoAd_00171]
SoAdSocketTcpTxQuota	0..1	[ECUC_SoAd_00142]
SoAdSocketTCPOptionFilterRef	0..1	[ECUC_SoAd_00155]
SoAdSocketTcpTlsConnectionRef	0..1	[ECUC_SoAd_00163]

No Included Containers

⌋

[ECUC_SoAd_00183] Definition of EcucBooleanParamDef SoAdSocketIfImmediateTxConfirmation ⌈

Parameter Name	SoAdSocketIfImmediateTxConfirmation	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	–	
Description	Controls immediate IF transmit confirmation for TCP sockets.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00174] Definition of EcucFloatParamDef SoAdSocketTcpAutoConnectTimeout ⌈

Parameter Name	SoAdSocketTcpAutoConnectTimeout	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Specifies the time in seconds how long 'TCP connect attempts are repeated to reach SOAD_SOCON_ONLINE. This parameter is restricted to socket connection groups which are initiating a TCP connection and are under control of SoAd. Dependency: Parameter shall be configured only if SoAdSocketConnectionGroup.SoAdSocketAutomaticSoConSetup and SoAdSocketConnectionGroup.SoAdSocketTcpInitiate are set to true.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00147] Definition of EcucBooleanParamDef SoAdSocketTcpImmediateTpTxConfirmation ⌈

Parameter Name	SoAdSocketTcpImmediateTpTxConfirmation	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	–	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00022] Definition of EcucBooleanParamDef SoAdSocketTcpInitiate ⌈

Parameter Name	SoAdSocketTcpInitiate	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00148] Definition of EcucBooleanParamDef SoAdSocketTcpKeepAlive ⌈

Parameter Name	SoAdSocketTcpKeepAlive	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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. Dependency: TcpIpTcpKeepAliveEnabled	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00152] Definition of EcucFloatParamDef SoAdSocketTcpKeepAliveInterval ⌈

Parameter Name	SoAdSocketTcpKeepAliveInterval	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Specifies the interval in seconds between subsequent keepalive probes. Dependency: SoAdSocketTcpKeepAlive	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00151] Definition of EcucIntegerParamDef SoAdSocketTcpKeepAliveProbesMax ⌈

Parameter Name	SoAdSocketTcpKeepAliveProbesMax	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Maximum number of times that TCP retransmits an individual data segment before aborting the connection. Dependency: SoAdSocketTcpKeepAlive	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00153] Definition of EcucFloatParamDef SoAdSocketTcpKeepAliveTime ⌈

Parameter Name	SoAdSocketTcpKeepAliveTime	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Specifies the time in seconds between the last data packet sent and the first keepalive probe. Dependency: SoAdSocketTcpKeepAlive	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	7200	
Post-Build Multiplicity	Loadable	Selectable

▽

△

Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00179] Definition of EcucIntegerParamDef SoAdSocketTcpMaxRetransmissions ⌈

Parameter Name	SoAdSocketTcpMaxRetransmissions	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	–	
Description	Maximum number of times that a TCP segment is retransmitted before the TCP connection is closed. This parameter overrules TcpIpTcpMaxRtx for this SoAdSocket ConnectionGroup. Note: This parameter also applies for FIN retransmissions.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967295	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00180] Definition of EcucFloatParamDef SoAdSocketTcpMaxRetransmissionTimeout ⌈

Parameter Name	SoAdSocketTcpMaxRetransmissionTimeout	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	–	
Description	Maximum value (clamp) of clamped exponential backoff timeout in [s] before an unacknowledged TCP segment is sent. This parameter overrules TcpIpTcpMax RetransmissionTimeout for this SoAdSocketConnectionGroup.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0.001 .. INF]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00023] Definition of EcucBooleanParamDef SoAdSocketTcpNoDelay ⌈

Parameter Name	SoAdSocketTcpNoDelay
Parent Container	SoAdSocketTcp
Upstream mapping defined	[12, CP_TPS_SystemTemplate]

▽

△

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 TcpIpTcp NagleEnabled is applied. Note: This parameter must not be set to FALSE if TcpIpTcp NagleEnabled is set to FALSE. Dependency: TcpIpTcpNagleEnabled	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00171] Definition of EcucFloatParamDef SoAdSocketTcpRetransmissionTimeout ⌈

Parameter Name	SoAdSocketTcpRetransmissionTimeout	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Timeout in [s] before an unacknowledged TCP segment is sent again. This parameter overrules TcpIpTcpRetransmissionTimeout for this SoAdSocketConnectionGroup. If this parameter is enabled together with a maximum like TcpIpTcpMaxRetransmission Timeout or SoAdSocketTcpMaxRetransmissionTimeout then SoAdSocketTcp RetransmissionTimeout is considered as initial value for first retransmission before the next valid acknowledgment arrives. Dependency: SoAdSocketTcpMaxRetransmissionTimeout, TcpIpTcpMax RetransmissionTimeout	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0.001 .. INF]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00142] Definition of EcucIntegerParamDef SoAdSocketTcpTxQuota ⌈

Parameter Name	SoAdSocketTcpTxQuota
Parent Container	SoAdSocketTcp
Upstream mapping defined	–
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 Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00155] Definition of EcucReferenceDef SoAdSocketTCPOptionFilterRef ⌈

Parameter Name	SoAdSocketTCPOptionFilterRef	
Parent Container	SoAdSocketTcp	
Upstream mapping defined	–	
Description	Specifies which TCP option filter shall be applied on the related socket.	
Multiplicity	0..1	
Type	Symbolic name reference to TcpIpTcpConfigOptionFilter	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00163] Definition of EcucReferenceDef SoAdSocketTcpTlsConnectionRef ⌈

Parameter Name	SoAdSocketTcpTlsConnectionRef
Parent Container	SoAdSocketTcp
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
Description	If set the TCP socket is assigned to a TLS connection. The SoAd need to call Tcpip_ChangeParameter with the reference to the TLS connection as the parameter.
Multiplicity	0..1
Type	Symbolic name reference to TcpIpTlsConnection
Pre-Compile only	true

⌋

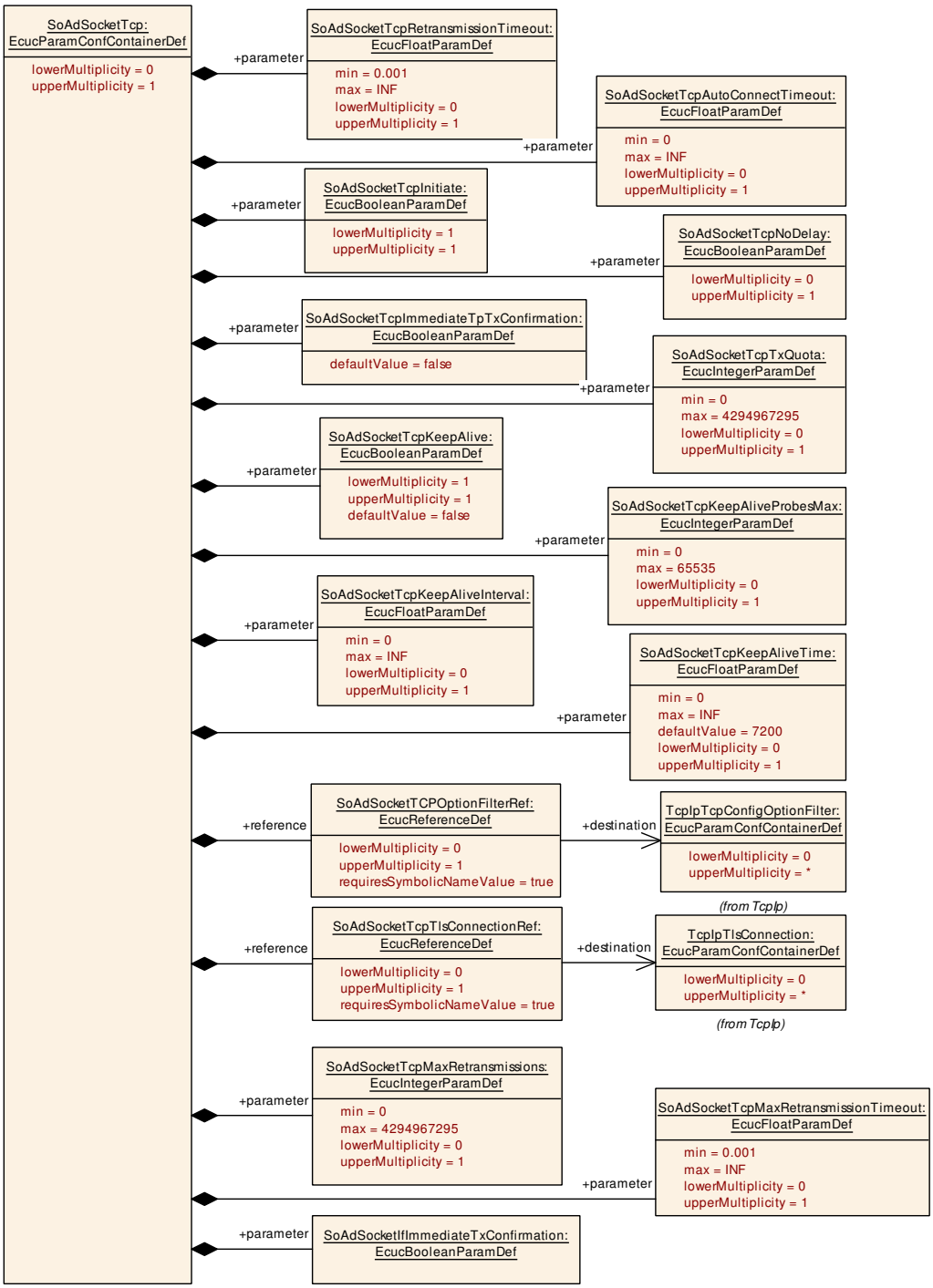


Figure 10.9: SoAd Socket Connection Protocol TCP container

10.2.11 SoAdSocketRemoteAddress

[ECUC_SoAd_00113] Definition of EcucParamConfContainerDef SoAdSocketRemoteAddress ⌈

Container Name	SoAdSocketRemoteAddress
Parent Container	SoAdSocketConnection
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
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().
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdSocketRemoteIpAddress	1	[ECUC_SoAd_00019]
SoAdSocketRemotePort	1	[ECUC_SoAd_00020]

No Included Containers

⌋

[ECUC_SoAd_00019] Definition of EcucStringParamDef SoAdSocketRemoteIp Address ⌈

Parameter Name	SoAdSocketRemoteIpAddress	
Parent Container	SoAdSocketRemoteAddress	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	IP address of remote node. The configured address must be of the same TcpIpDomain Type (i.e. IPv4 or IPv6) as the TcpIpLocalAddr referred by SoAdSocketLocalAddress Ref . To accept any remote IP address, set SoAdSocketRemoteIpAddress to "ANY". See message acceptance policy for more details. Dependency: TcpIpDomainType	
Multiplicity	1	
Type	EcucStringParamDef	
Default value	–	
Regular Expression	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00020] Definition of EcucIntegerParamDef SoAdSocketRemote Port ⌈

Parameter Name	SoAdSocketRemotePort	
Parent Container	SoAdSocketRemoteAddress	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Remote UDP or TCP port used for this connection. To accept any remote port, set So AdSocketRemotePort to 0. See message acceptance policy for more details.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	

▽

△

Post-Build Value	Loadable	Selectable

⌋

10.2.12 SoAdSocketRoute

[ECUC_SoAd_00008] Definition of EcucParamConfContainerDef SoAdSocketRoute ⌈

Container Name	SoAdSocketRoute
Parent Container	SoAdConfig
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
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.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdRxPduHeaderId	0..1	[ECUC_SoAd_00036]
SoAdRxSocketConnOrSocketConnBundleRef	1	[ECUC_SoAd_00035]

Included Containers		
Container Name	**Multiplicity**	**Description**
SoAdSocketRouteDest	1..*	Container which specifies the socket route destination.

⌋

[ECUC_SoAd_00036] Definition of EcucIntegerParamDef SoAdRxPduHeaderId ⌈

Parameter Name	SoAdRxPduHeaderId	
Parent Container	SoAdSocketRoute	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	ID contained in the packet received on the TCP/IP connection if the PDU header option is enabled. Dependency: Parameter shall be configured for SoAdSocketRoute that is related to a SoAdSocketConnection belonging to a SoAdSocketConnectionGroup with SoAdPdu HeaderEnable set to TRUE.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967296	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00035] Definition of EcucChoiceReferenceDef SoAdRxSocketConnOrSocketConnBundleRef ⌈

Parameter Name	SoAdRxSocketConnOrSocketConnBundleRef	
Parent Container	SoAdSocketRoute	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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. Dependency: SoAdRxUpperLayerType	
Multiplicity	1	
Type	Choice reference to [SoAdSocketConnection, SoAdSocketConnectionGroup]	
Post-Build Value	Loadable	Selectable

⌋

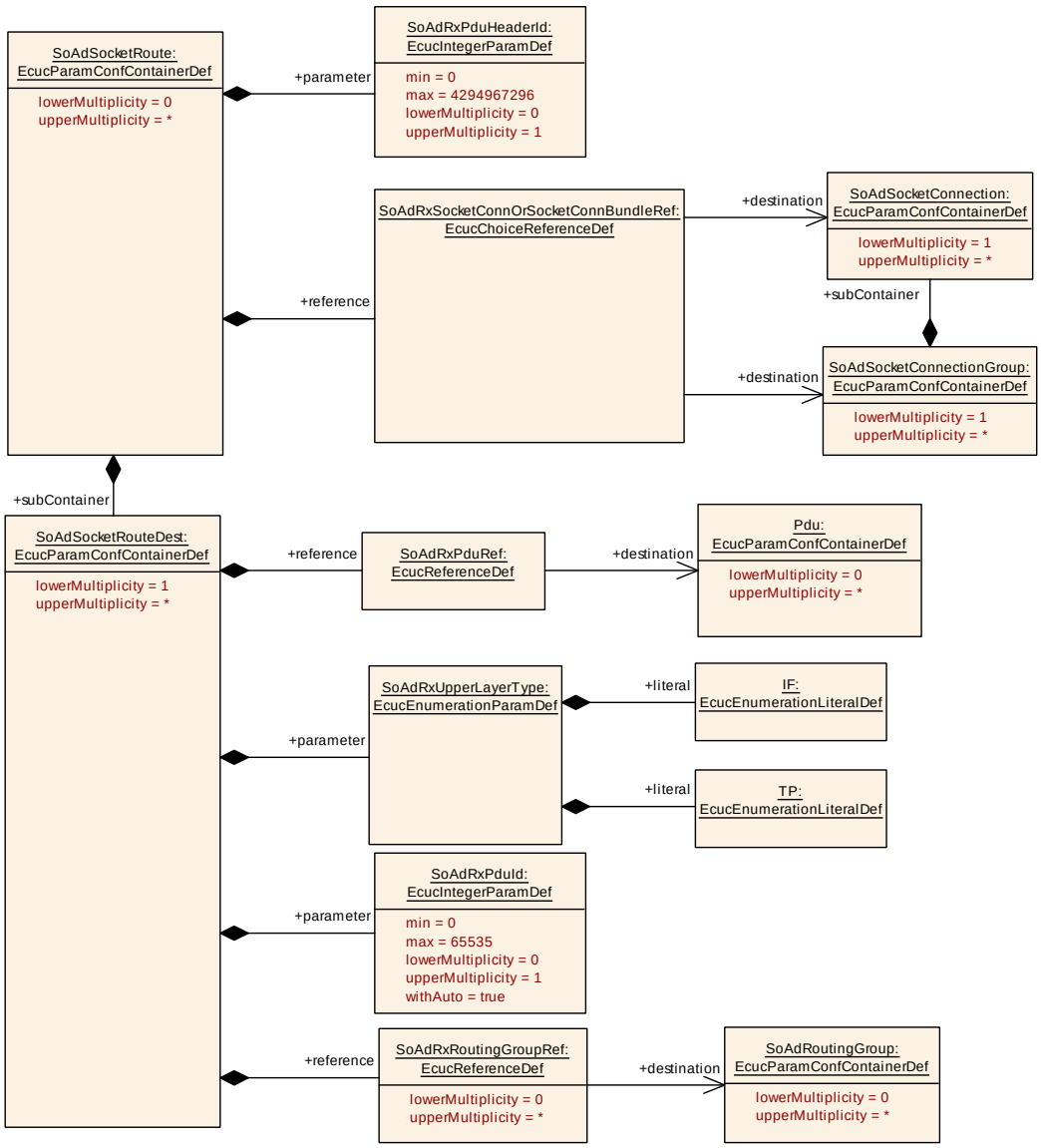


Figure 10.10: SoAd Socket Route container

10.2.13 SoAdSocketRouteDest

[ECUC_SoAd_00114] Definition of EcucParamConfContainerDef SoAdSocketRouteDest ⌈

Container Name	SoAdSocketRouteDest
Parent Container	SoAdSocketRoute
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
Description	Describes the upper layer destination PDU for a message received on a TcpIp socket. This PDU can produce meta data items of type SOCKET_CONNECTION_ID_16. Multiple socket route destinations in the SoAdSocketRoute can only be used for upper layers of interface type (IF) and only for SoAdSocketRoute referring a Socket ConnectionGroup. In this case SoAdRoutingGroups shall be used to map each SoAd SocketRouteDest uniquely to different socket connections of the SocketConnection Group.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdRxPduId	0..1	[ECUC_SoAd_00116]
SoAdRxUpperLayerType	1	[ECUC_SoAd_00115]
SoAdRxPduRef	1	[ECUC_SoAd_00038]
SoAdRxRoutingGroupRef	0..*	[ECUC_SoAd_00117]

No Included Containers

⌋

[ECUC_SoAd_00116] Definition of EcucIntegerParamDef SoAdRxPduId ⌈

Parameter Name	SoAdRxPduId	
Parent Container	SoAdSocketRouteDest	
Upstream mapping defined	–	
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	–	
With Auto	true	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00115] Definition of EcucEnumerationParamDef SoAdRxUpperLayerType ⌈

Parameter Name	SoAdRxUpperLayerType	
Parent Container	SoAdSocketRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00038] Definition of EcucReferenceDef SoAdRxPduRef ⌈

Parameter Name	SoAdRxPduRef	
Parent Container	SoAdSocketRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Reference to the global PDU structure	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00117] Definition of EcucReferenceDef SoAdRxRoutingGroupRef ⌈

Parameter Name	SoAdRxRoutingGroupRef	
Parent Container	SoAdSocketRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Reference to the routing group. Mandatory if the parent SoAdSocketRoute contains more than one SoAdSocketRouteDest. Dependency: The same SoAdRoutingGroup shall not be referenced by multiple SoAd SocketRouteDest of the same SoAdSocketRoute.	
Multiplicity	0..*	
Type	Reference to SoAdRoutingGroup	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

10.2.14 SoAdPduRoute

[ECUC_SoAd_00007] Definition of EcucParamConfContainerDef SoAdPduRoute ⌈

Container Name	SoAdPduRoute
Parent Container	SoAdConfig
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
Description	Describes the path of a PDU from an upper layer of the SoAd to the socket in the TCP/IP stack for transmission. This PDU can consume meta data items of type SOCKET_CONNECTION_ID_16.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdTxPduCollectionSemantics	1	[ECUC_SoAd_00160]
SoAdTxPduId	1	[ECUC_SoAd_00031]
SoAdTxUpperLayerType	1	[ECUC_SoAd_00118]
SoAdTxPduRef	1	[ECUC_SoAd_00030]

Included Containers		
Container Name	**Multiplicity**	**Description**
SoAdPduRouteDest	1..*	Container which specifies the PDU route destination.

⌋

[ECUC_SoAd_00160] Definition of EcucEnumerationParamDef SoAdTxPduCollectionSemantics ⌈

Parameter Name	SoAdTxPduCollectionSemantics	
Parent Container	SoAdPduRoute	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Specifies if this PDU shall be collected using a queued or last-is-best semantics. This parameter is only relevant if the PDU collection feature is enabled. Shall only be set to SOAD_COLLECT_LAST_IS_BEST if the related upper layer is configured with SoAdIf TriggerTransmit set to TRUE. Dependency: SoAdSocketnPduUdpTxBufferMin, SoAdIfTriggerTransmit	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	SOAD_COLLECT_LAST_IS_BEST	The PDU data will be fetched via <Up>_[SoAd][If]TriggerTransmit just before the transmission executes.
	SOAD_COLLECT_QUEUED	The PDU data will instantly be stored in the context of the SoAd_IfTransmit API.
Default value	SOAD_COLLECT_QUEUED	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00031] Definition of EcucIntegerParamDef SoAdTxPduId ⌈

Parameter Name	SoAdTxPduId	
Parent Container	SoAdPduRoute	
Upstream mapping defined	–	
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_SoAd_00118] Definition of EcucEnumerationParamDef SoAdTxUpperLayerType ⌈

Parameter Name	SoAdTxUpperLayerType	
Parent Container	SoAdPduRoute	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00030] Definition of EcucReferenceDef SoAdTxPduRef ⌈

Parameter Name	SoAdTxPduRef	
Parent Container	SoAdPduRoute	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Reference to the global PDU structure	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

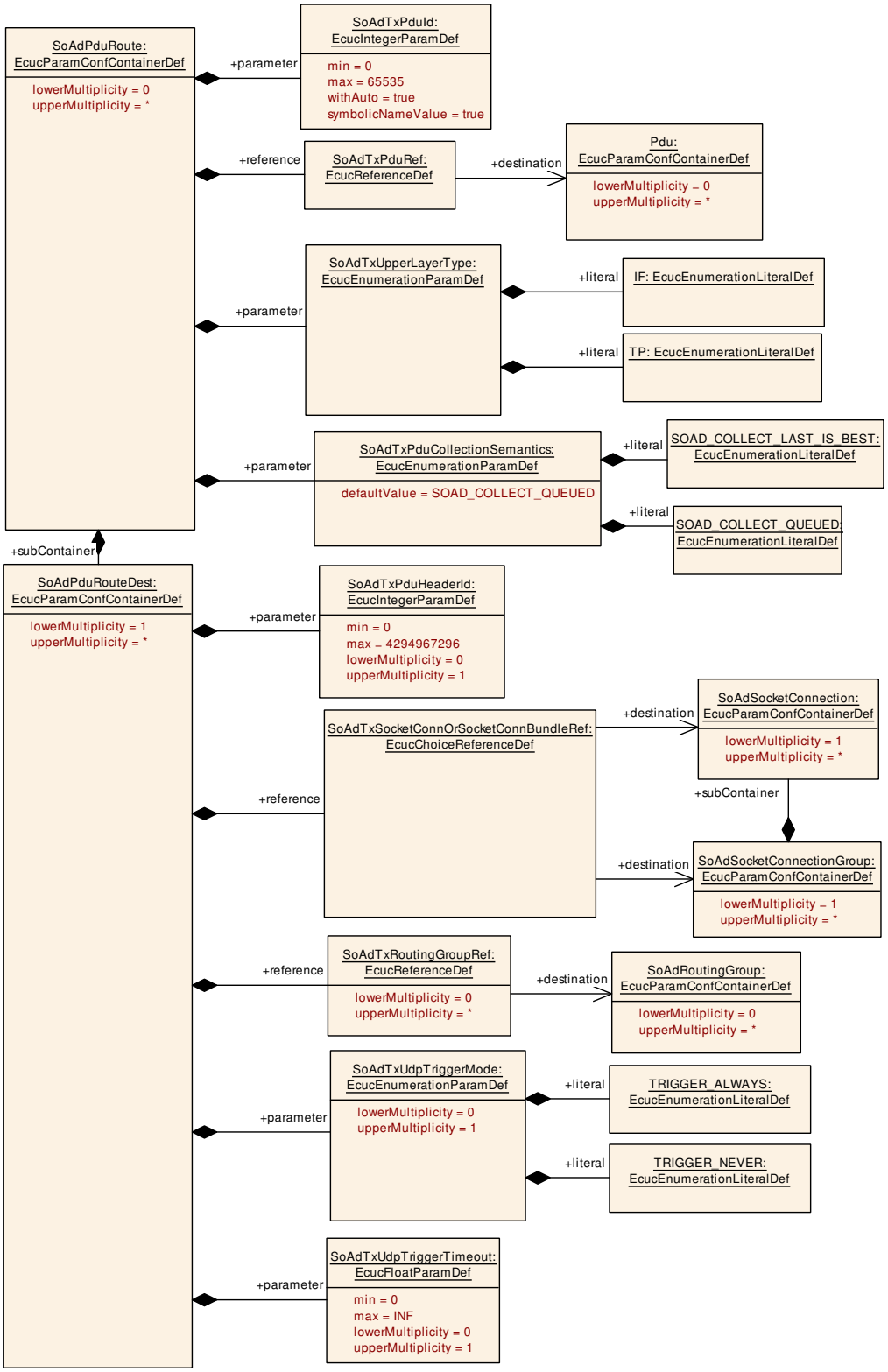


Figure 10.11: SoAd Pdu Route container

10.2.15 SoAdPduRouteDest

[ECUC_SoAd_00119] Definition of EcucParamConfContainerDef SoAdPduRouteDest ⌈

Container Name	SoAdPduRouteDest
Parent Container	SoAdPduRoute
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
Description	Specifies the PDU route destination.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdTxPduHeaderId	0..1	[ECUC_SoAd_00120]
SoAdTxUdpTriggerMode	0..1	[ECUC_SoAd_00136]
SoAdTxUdpTriggerTimeout	0..1	[ECUC_SoAd_00150]
SoAdTxRoutingGroupRef	0..*	[ECUC_SoAd_00123]
SoAdTxSocketConnOrSocketConnBundleRef	1	[ECUC_SoAd_00034]

No Included Containers

⌋

[ECUC_SoAd_00120] Definition of EcucIntegerParamDef SoAdTxPduHeaderId ⌈

Parameter Name	SoAdTxPduHeaderId	
Parent Container	SoAdPduRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	ID to be sent on the TCP/IP connection if the PDU header option is enabled. Dependency: Parameter shall be configured for SoAdPduRouteDest that is related to a SoAdSocketConnection belonging to a SoAdSocketConnectionGroup with SoAdPduHeaderEnable set to TRUE.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 4294967296	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00136] Definition of EcucEnumerationParamDef SoAdTxUdpTriggerMode ⌈

Parameter Name	SoAdTxUdpTriggerMode
Parent Container	SoAdPduRouteDest
Upstream mapping defined	[12, CP_TPS_SystemTemplate]

▽

△

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. Dependency: SoAdSocketProtocol, SoAdTxUpperLayerType	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	TRIGGER_ALWAYS	PDU triggers the transmission
	TRIGGER_NEVER	PDU does not trigger the transmission
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00150] Definition of EcucFloatParamDef SoAdTxUdpTriggerTimeout ⌈

Parameter Name	SoAdTxUdpTriggerTimeout	
Parent Container	SoAdPduRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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 SoAdTxUdp TriggerMode is TRIGGER_NEVER. Dependency: SoAdTxUdpTriggerMode	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. INF]	
Default value	–	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00123] Definition of EcucReferenceDef SoAdTxRoutingGroupRef ⌈

Parameter Name	SoAdTxRoutingGroupRef	
Parent Container	SoAdPduRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
Description	Reference to the routing group.	
Multiplicity	0..*	
Type	Reference to SoAdRoutingGroup	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_SoAd_00034] Definition of EcucChoiceReferenceDef SoAdTxSocketConnOrSocketConnBundleRef ⌈

Parameter Name	SoAdTxSocketConnOrSocketConnBundleRef	
Parent Container	SoAdPduRouteDest	
Upstream mapping defined	[12, CP_TPS_SystemTemplate]	
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. Dependency: SoAdTxUpperLayerType	
Multiplicity	1	
Type	Choice reference to [SoAdSocketConnection, SoAdSocketConnectionGroup]	
Post-Build Value	Loadable	Selectable

⌋

10.2.16 SoAdRoutingGroup

[ECUC_SoAd_00109] Definition of EcucParamConfContainerDef SoAdRoutingGroup ⌈

Container Name	SoAdRoutingGroup
Parent Container	SoAdConfig
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
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 SoAd PduRoute or the other way around specified by a SoAdSocketRoute.
Multiplicity	0..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SoAdRoutingGroupId	1	[ECUC_SoAd_00121]
SoAdRoutingGroupIsEnabledAtInit	1	[ECUC_SoAd_00122]
SoAdRoutingGroupTxTriggerable	1	[ECUC_SoAd_00146]

No Included Containers

⌋

[ECUC_SoAd_00121] Definition of EcucIntegerParamDef SoAdRoutingGroupId ⌈

Parameter Name	SoAdRoutingGroupId
Parent Container	SoAdRoutingGroup
Upstream mapping defined	–
Description	Unique ID of Routing Group

▽

△

Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_SoAd_00122] Definition of EcucBooleanParamDef SoAdRoutingGroupIsEnabledAtInit ⌈

Parameter Name	SoAdRoutingGroupIsEnabledAtInit	
Parent Container	SoAdRoutingGroup	
Upstream mapping defined	–	
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 Value	Loadable	Selectable

⌋

[ECUC_SoAd_00146] Definition of EcucBooleanParamDef SoAdRoutingGroupTxTriggerable ⌈

Parameter Name	SoAdRoutingGroupTxTriggerable
Parent Container	SoAdRoutingGroup
Upstream mapping defined	[12, CP_TPS_SystemTemplate]
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
Pre-Compile only	true

⌋

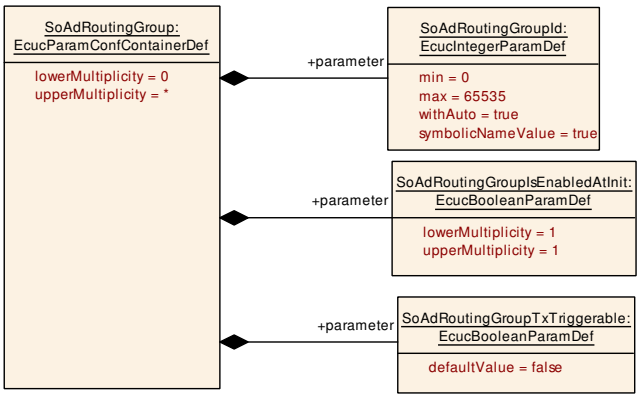


Figure 10.12: SoAd Routing Group container

10.3 Configuration constraints

[SWS_SoAd_CONSTR_00001] Configuration of SoAdKeepLocalPduBuffer ⌈SoAdKeepLocalPduBuffer shall be configurable at `SoAdSocketConnectionGroup` and shall control whether SoAd applies local-buffer retention semantics for the mapped communication flow.⌋

[SWS_SoAd_CONSTR_00002] Alignment validation for KeepLocalPduBuffer feature ⌈If SoAdKeepLocalPduBuffer is set to `TRUE`, alignment shall be validated at configuration time by tracing `SoAdSocketLocalAddressRef` to the referenced TcpIp controller and checking `PduBufferAlignment` values assigned to that controller via `TcpIpCtrlRxPdu` and `TcpIpCtrlTxPdu`.⌋

[SWS_SoAd_CONSTR_00003] Alignment constraint for upper-layer transmit buffer ⌈If `KeepLocalPduBuffer` is set to `TRUE`, the data buffer provided by upper layers for transmission via SoAd shall satisfy the configured `PduBufferAlignment` of the referenced global PDU. This shall be treated as a constraint on feature usage and configuration, not as a SoAd-internal functional realignment step.⌋

[SWS_SoAd_CONSTR_00004] Synchronous copy behavior when KeepLocalPduBuffer is disabled ⌈If `KeepLocalPduBuffer` is set to `FALSE` for an IF PDU, SoAd shall use synchronous copy processing in caller context for transmission requests and reception indications. For such PDUs, no local-buffer retention semantics shall be required.⌋

[SWS_SoAd_CONSTR_00005] IF-PDU whitelist scope for KeepLocalPduBuffer feature behavior ⌈The feature behavior associated with `KeepLocalPduBuffer` (local-buffer retention semantics and asynchronous-copy related copy-path handling) shall apply only to IF-PDU based communication paths.⌋

[SWS_SoAd_CONSTR_00006] Configuration of immediate IF transmit confirmation behavior ⌈`SoAdSocketIfImmediateTxConfirmation` shall be configurable

for TCP socket connections and shall control whether SoAd configures immediate IF transmit confirmation behavior towards TcpIp according to [SWS_SoAd_00689].⌋

[SWS_SoAd_CONSTR_00007] DTLS: AutomaticSoConSetup conflict guard ⌈For a UDP socket connection group with `SoAdDatagramTlsConnectionRef` set, `SoAdSocketAutomaticSoConSetup` shall be set to `FALSE`.⌋

Note: With this constraint, remote-address assignment is handled through the manually-set remote-address path (for example via `SoAd_SetRemoteAddr`() or `SoAd_SetUniqueRemoteAddr`()).

[SWS_SoAd_CONSTR_00008] DTLS: TCP/DTLS protocol mismatch guard ⌈For a TCP socket connection with `SoAdSocketTcpTlsConnectionRef` set, the referenced `TcpIpTlsConnection` shall not contain `TcpIpDtlsInstance`.⌋

[SWS_SoAd_CONSTR_00009] SoAdDatagramTlsConnectionRef dependency ⌈`SoAdDatagramTlsConnectionRef` shall reference a `TcpIpTlsConnection` used for DTLS over UDP. For a UDP connection with `SoAdDatagramTlsConnectionRef` set and referencing a `TcpIpTlsConnection` with DTLS disabled, the configuration shall be rejected.⌋

[SWS_SoAd_CONSTR_00010] DTLS instance count check ⌈For a UDP socket connection group with `SoAdDatagramTlsConnectionRef` set to a `TcpIpTlsConnection` whose `TcpIpTlsProtocol` selects a DTLS configuration, the number of SoAd socket connections that reference this `TcpIpTlsConnection` shall have the same value as the number of configured DTLS instances in that referenced `TcpIpTlsConnection`.⌋

[SWS_SoAd_CONSTR_00011] DTLS TlsConnection single reference ⌈For a UDP socket connection group with `SoAdDatagramTlsConnectionRef` set to a `TcpIpTlsConnection` whose `TcpIpTlsProtocol` selects a DTLS configuration, if that same `TcpIpTlsConnection` is referenced by another `SoAdSocketConnectionGroup` via `SoAdDatagramTlsConnectionRef`, the configuration shall be rejected.⌋

10.4 Published Information

For details refer to [2] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[SWS_SoAd_NA_00296]

Upstream requirements: 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_00491, SRS_BSW_00490, SRS_BSW_00478, SRS_BSW_00472, SRS_BSW_00471, SRS_BSW_00469, SRS_BSW_00461, SRS_BSW_00422, SRS_BSW_00419, SRS_BSW_00403, SRS_BSW_00395, SRS_BSW_00393, SRS_BSW_00392, SRS_BSW_00390, SRS_BSW_00389, SRS_BSW_00388, SRS_BSW_00386, SRS_BSW_00384, SRS_BSW_00383, SRS_BSW_00369, SRS_BSW_00004

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

[SWS_SoAd_00771] [SWS_SoAd_00772] [SWS_SoAd_00773] [SWS_SoAd_00774] [SWS_SoAd_00775] [SWS_SoAd_00776]

B.1.2 Changed Specification Items in R23-11

[SWS_SoAd_00197] [SWS_SoAd_00559] [SWS_SoAd_00563] [SWS_SoAd_00692] [SWS_SoAd_00764]

B.1.3 Deleted Specification Items in R23-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R24-11

B.2.1 Added Specification Items in R24-11

[ECUC_SoAd_00178] [SWS_SoAd_00777]

B.2.2 Changed Specification Items in R24-11

[ECUC_SoAd_00003] [ECUC_SoAd_00008] [ECUC_SoAd_00130] [ECUC_SoAd_00168] [SWS_SoAd_00180] [SWS_SoAd_00181] [SWS_SoAd_00504] [SWS_SoAd_00689] [SWS_SoAd_00692] [SWS_SoAd_00764] [SWS_SoAd_00772] [SWS_SoAd_00773] [SWS_SoAd_00774] [SWS_SoAd_00775]

B.2.3 Deleted Specification Items in R24-11

[ECUC_SoAd_00176] [ECUC_SoAd_00177]

B.3 Traceable item history of this document according to AUTOSAR Release R25-11

B.3.1 Added Specification Items in R25-11

[ECUC_SoAd_00179] [ECUC_SoAd_00180]

B.3.2 Changed Specification Items in R25-11

[ECUC_SoAd_00171] [SWS_SoAd_00689]

B.3.3 Deleted Specification Items in R25-11

[ECUC_SoAd_00156]

B.4 Traceable item history of this document according to AUTOSAR Release R26-11

B.4.1 Added Specification Items in R26-11

[ECUC_SoAd_00181] [ECUC_SoAd_00183] [SWS_SoAd_00778] [SWS_SoAd_00779] [SWS_SoAd_00780] [SWS_SoAd_00781] [SWS_SoAd_00782] [SWS_SoAd_00783] [SWS_SoAd_00784] [SWS_SoAd_00785] [SWS_SoAd_00786] [SWS_SoAd_00787] [SWS_SoAd_00788] [SWS_SoAd_00789] [SWS_SoAd_00790] [SWS_SoAd_00791] [SWS_SoAd_00793] [SWS_SoAd_00794] [SWS_SoAd_00795] [SWS_SoAd_00796] [SWS_SoAd_00797] [SWS_SoAd_00798] [SWS_SoAd_00799] [SWS_SoAd_00800] [SWS_SoAd_00801] [SWS_SoAd_00802] [SWS_SoAd_00803] [SWS_SoAd_91012]

B.4.2 Changed Specification Items in R26-11

[ECUC_SoAd_00001] [ECUC_SoAd_00003] [ECUC_SoAd_00019] [ECUC_SoAd_00023] [ECUC_SoAd_00024] [ECUC_SoAd_00034] [ECUC_SoAd_00035] [ECUC_SoAd_00036] [ECUC_SoAd_00117] [ECUC_SoAd_00120] [ECUC_SoAd_00133] [ECUC_SoAd_00135] [ECUC_SoAd_00136] [ECUC_SoAd_00137] [ECUC_SoAd_00140] [ECUC_SoAd_00141] [ECUC_SoAd_00148] [ECUC_SoAd_00150] [ECUC_SoAd_00151] [ECUC_SoAd_00152] [ECUC_SoAd_00153] [ECUC_SoAd_00154]

[ECUC_SoAd_00160] [ECUC_SoAd_00171] [ECUC_SoAd_00174] [SWS_SoAd_00097] [SWS_SoAd_00107] [SWS_SoAd_00146] [SWS_SoAd_00180] [SWS_SoAd_00181] [SWS_SoAd_00503] [SWS_SoAd_00591] [SWS_SoAd_00689] [SWS_SoAd_00692] [SWS_SoAd_00763] [SWS_SoAd_00764]

B.4.3 Deleted Specification Items in R26-11

[ECUC_SoAd_00004] [SWS_SoAd_00096] [SWS_SoAd_00197]

B.4.4 Added Constraints in R26-11

[SWS_SoAd_CONSTR_00001] [SWS_SoAd_CONSTR_00002] [SWS_SoAd_CONSTR_00003] [SWS_SoAd_CONSTR_00004] [SWS_SoAd_CONSTR_00005] [SWS_SoAd_CONSTR_00006] [SWS_SoAd_CONSTR_00007] [SWS_SoAd_CONSTR_00008] [SWS_SoAd_CONSTR_00009] [SWS_SoAd_CONSTR_00010] [SWS_SoAd_CONSTR_00011]

B.4.5 Changed Constraints in R26-11

none

B.4.6 Deleted Constraints in R26-11

none