

<b>Document Title</b>	Specification of Service Discovery
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	616

<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R23-11

Document Change History			
Date	Release	Changed by	Description
2023-11-23	R23-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Added Secure SOME/IP-ACL</li> <li>Minor bugfixes and editorial changes</li> </ul>
2022-11-24	R22-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>TTL for FindService enteries</li> <li>Minor bugfixes and editorial changes</li> </ul>
2021-11-25	R21-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Introduced optional functionality to subscribe to a multicast address pre-defined by a ClientService</li> <li>Consideration of the connection status of a security associations for clients and servers was added</li> <li>Harmonization of <i>Specification of Service Discovery</i> and <i>Service Discovery Protocol specification</i>: <ul style="list-style-type: none"> <li>removal of duplicate specification items</li> <li>moving of specification items from <i>Specification of Service Discovery</i> to <i>Service Discovery Protocol specification</i></li> </ul> </li> <li>Minor bugfixes and editorial changes</li> </ul>





2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Alignments with Service Discovery Protocol specification</li> <li>• Several minor bugfixes</li> <li>• Editorial changes</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Service activation depending on PNCs</li> <li>• Retry mechanism in combination with Cyclic Offers</li> <li>• EventGroup subscription updates from different servers</li> <li>• Clarification of SubscribeEventgroupNack handling</li> <li>• Changed Document Status from Final to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Retry subscription feature added</li> <li>• Load Balancing Option added</li> <li>• Minor bugfixes</li> </ul>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Several minor bugfixes</li> <li>• Editorial changes</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Major improvement (SoAd interaction)</li> <li>• Several bugfixes</li> <li>• Editorial changes</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Debugging support marked as obsolete</li> <li>• Clarifications</li> <li>• Minor bugfixes</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Fixed Service Migration support at client side</li> <li>• Support for more efficient SoAd interface</li> <li>• Optimized StopSubscribe/Subscribe load</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Editorial changes</li> <li>• More detailed endpoint handling</li> <li>• More detailed message building</li> </ul>





2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• No major changes have been made</li><li>• Editorial changes</li><li>• Removed chapter(s) on change documentation</li></ul>
2013-03-15	4.1.1	AUTOSAR Release Management	<ul style="list-style-type: none"><li>• Initial Release</li></ul>

## 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.

## Contents

1	Introduction and functional overview	10
2	Acronyms, Abbreviations and Definitions	11
3	Related documentation	13
3.1	Input documents & related standards and norms	13
3.2	Related specification	13
4	Constraints and assumptions	14
4.1	Limitations	14
4.2	Limitations for communication with Adaptive Platform	14
4.3	Applicability to car domains	14
5	Dependencies to other modules	15
5.1	AUTOSAR BSW Scheduler	15
5.2	AUTOSAR BSW Mode Manager	15
5.3	AUTOSAR Socked Adaptor	15
5.4	AUTOSAR Default Error Tracer	15
5.5	AUTOSAR Diagnostic Event Manager	16
5.6	AUTOSAR Non Volatile Memory	16
5.7	File structure	16
5.7.1	Code file structure	16
5.7.2	Header file structure	16
6	Requirements Tracing	17
7	Functional specification	18
7.1	Background & Rationale	18
7.2	Requirements	20
7.2.1	General requirements	20
7.2.2	Ethernet Communication	22
7.2.3	State Handling	23
7.2.4	Interaction with Socket Adaptor	24
7.2.5	Subscribe Eventgroup retry handling	25
7.3	Message format	27
7.3.1	Request ID	28
7.3.2	Protocol Version field	29
7.3.3	Interface Version field	29
7.3.4	Message Type field	29
7.3.5	Return Code field	29
7.3.6	Flags field	29
7.3.7	Reserved field	30
7.3.8	Entries Array	30
7.3.8.1	Entry Format Type 1	30
7.3.8.2	Entry Format Type 2	31

7.3.9	Options Array . . . . .	32
7.3.9.1	Configuration Option . . . . .	32
7.3.9.2	IPv4 Endpoint Option . . . . .	33
7.3.9.3	IPv6 Endpoint Option . . . . .	34
7.3.9.4	IPv4 Multicast Option . . . . .	35
7.3.9.5	IPv6 Multicast Option . . . . .	35
7.3.9.6	IPv4 SD Endpoint Option . . . . .	36
7.3.9.7	IPv6 SD Endpoint Option . . . . .	36
7.3.9.8	Handling missing, redundant, and conflicting Options . . . . .	37
7.3.9.9	Security considerations for Options . . . . .	38
7.3.10	Entries referencing Options . . . . .	38
7.4	Service Discovery Entry Types . . . . .	40
7.4.1	Entries for Services (common requirements) . . . . .	40
7.4.2	FindService entry . . . . .	40
7.4.3	OfferService entry . . . . .	41
7.4.4	Building OfferService entries . . . . .	42
7.4.5	StopOfferService entry . . . . .	43
7.4.6	Eventgroup Entries (Common requirements) . . . . .	43
7.4.7	SubscribeEventgroup entry . . . . .	43
7.4.8	StopSubscribeEventgroup entry . . . . .	44
7.4.9	SubscribeEventgroupAck entry . . . . .	44
7.4.10	SubscribeEventgroupNack entry . . . . .	44
7.4.11	Building SubscribeEventgroup entries . . . . .	45
7.5	Sending and Receiving of Messages . . . . .	46
7.5.1	Sequence for message transmission . . . . .	47
7.5.2	Sequence for message reception . . . . .	47
7.5.3	Receiving Entries . . . . .	48
7.5.3.1	Answering behaviour, if receiving Service Discovery Entries via Multicast address . . . . .	49
7.6	Timings and repetitions for Server Service and Event Handlers . . . . .	51
7.6.1	Initial Wait Phase for Server Services . . . . .	51
7.6.2	Repetition Phase for Server Services . . . . .	53
7.6.3	Main Phase for Server Services . . . . .	56
7.6.4	Fan out control . . . . .	58
7.6.5	Sharing of SdServerTimer . . . . .	62
7.7	Timings and repetitions for Client Service and Consumed Eventgroups . . . . .	63
7.7.1	Down Phase for Client Services . . . . .	63
7.7.2	Initial Wait Phase for Client Services . . . . .	64
7.7.3	Repetition Phase for Client Services . . . . .	65
7.7.4	Main Phase for Client Services . . . . .	67
7.7.5	Fan in control . . . . .	73
7.7.6	Sharing of SdClientTimer . . . . .	74
7.8	Handling of SdServiceGroupS . . . . .	74
7.8.1	SdServiceGroup definitions . . . . .	74
7.8.1.1	Initialization of SdServiceGroupS . . . . .	76
7.8.1.2	Starting of SdServiceGroupS . . . . .	76

7.8.1.3	Stopping of SdServiceGroupS	76
7.9	SOME/IP-ACL	76
7.9.1	ACL Configuration	77
7.9.1.1	ACL update	77
7.9.2	ACL Policy Check	78
7.9.2.1	Client ACL	78
7.9.2.2	Server ACL	79
7.10	Security Events	81
7.11	Error Classification	81
7.11.1	Development Errors	82
7.11.2	Runtime Errors	82
7.11.3	Transient Faults	82
7.11.4	Production Errors	82
7.11.5	Extended Production Errors	83
7.11.5.1	SD_E_OUT_OF_RES	83
7.11.5.2	SD_E_MALFORMED_MSG	83
7.11.5.3	SD_E_SUBSCR_NACK_RECV	84
8	API specification	85
8.1	Imported types	85
8.2	Type definitions	85
8.2.1	Sd_ConfigType	85
8.2.2	Sd_ServerServiceSetStateType	86
8.2.3	Sd_ClientServiceSetStateType	86
8.2.4	Sd_ConsumedEventGroupSetStateType	86
8.2.5	Sd_ClientServiceCurrentStateType	87
8.2.6	Sd_ConsumedEventGroupCurrentStateType	87
8.2.7	Sd_EventHandlerCurrentStateType	87
8.2.8	Sd_ConfigOptionStringType	88
8.2.9	Sd_ServiceGroupIdType	88
8.2.10	Sd_ServiceAclUpdateType	88
8.3	Function definitions	89
8.3.1	Sd_Init	89
8.3.2	Sd_GetVersionInfo	89
8.3.3	Sd_ServerServiceSetState	90
8.3.4	Sd_ClientServiceSetState	91
8.3.5	Sd_ConsumedEventGroupSetState	92
8.3.6	Sd_LocalIpAddrAssignmentChg	93
8.3.7	Sd_SoConModeChg	93
8.3.8	Sd_ServiceGroupStart	94
8.3.9	Sd_ServiceGroupStop	94
8.3.10	Sd_AclUpdate	95
8.3.11	Sd_AclPolicyCheck	96
8.4	Callback notifications	96
8.4.1	Sd_RxIndication	96
8.5	Scheduled functions	97

8.5.1	Sd_MainFunction	97
8.6	Expected interfaces	98
8.6.1	Mandatory Interfaces	98
8.6.2	Optional Interfaces	99
8.6.3	Configurable Interfaces	100
8.6.3.1	Sd_CapabilityRecordMatchCallout	100
9	Sequence diagrams	101
9.1	CLIENT / SERVER: Sd_RxIndication	101
9.2	SERVER: Response Behavior	102
9.3	CLIENT: Response Behavior	103
9.4	SERVER: buildOfferServiceEntry	104
9.5	CLIENT: buildSubscribeEventgroupEntry	105
9.6	SERVER: buildSubscribeEventgroupAckEntry	106
9.7	CLIENT / SERVER: TransmitSdMessage	106
9.8	SERVER: AddClientToFanOut	107
9.9	SERVER: Start	108
9.10	CLIENT: Start	109
9.11	ACL: Service Offer	110
9.12	ACL: SubscribeEventgroup	110
9.13	ACL: Method call request	111
10	Configuration specification	112
10.1	How to read this chapter	112
10.2	Containers and configuration parameters	112
10.2.1	Sd	113
10.2.2	SdGeneral	115
10.2.3	SdConfig	120
10.2.4	SdCapabilityRecordMatchCallout	121
10.2.5	SdServiceGroup	122
10.2.6	SdInstance	123
10.2.7	SdClientService	125
10.2.8	SdBlocklistedVersions	133
10.2.9	SdClientCapabilityRecord	133
10.2.10	SdConsumedEventGroup	135
10.2.11	SdConsumedMethods	139
10.2.12	SdClientTimer	139
10.2.13	SdInstanceDemEventParameterRefs	144
10.2.14	SdInstanceMulticastRxPdu	146
10.2.15	SdInstanceTxPdu	147
10.2.16	SdInstanceUnicastRxPdu	147
10.2.17	SdServerService	148
10.2.18	SdEventHandler	155
10.2.19	SdEventHandlerMulticast	158
10.2.20	SdEventHandlerTcp	160
10.2.21	SdEventHandlerUdp	161
10.2.22	SdProvidedMethods	162



10.2.23	SdServerCapabilityRecord	162
10.2.24	SdServerTimer	163
10.2.25	SdServerServiceAllowedConsumers	166
10.2.26	SdClientServiceAllowedProvider	167
10.3	Published Information	169
A	Change history of AUTOSAR traceable items	170
A.1	Traceable item history of this document according to AUTOSAR Release R23-11	170
A.1.1	Added Specification Items in R23-11	170
A.1.2	Changed Specification Items in R23-11	170
A.1.3	Deleted Specification Items in R23-11	170

## 1 Introduction and functional overview

The AUTOSAR Service Discovery module offers functionality to detect and offer available services - i.e. functional entities - within the vehicle network. To do so, it makes use of the IP Multicast and so called SOME/IP-SD messages.

The Service Discovery module (Sd) is located between the AUTOSAR BSW Mode Manager module (BswM) and the AUTOSAR Socket Adaptor module (SoAd).

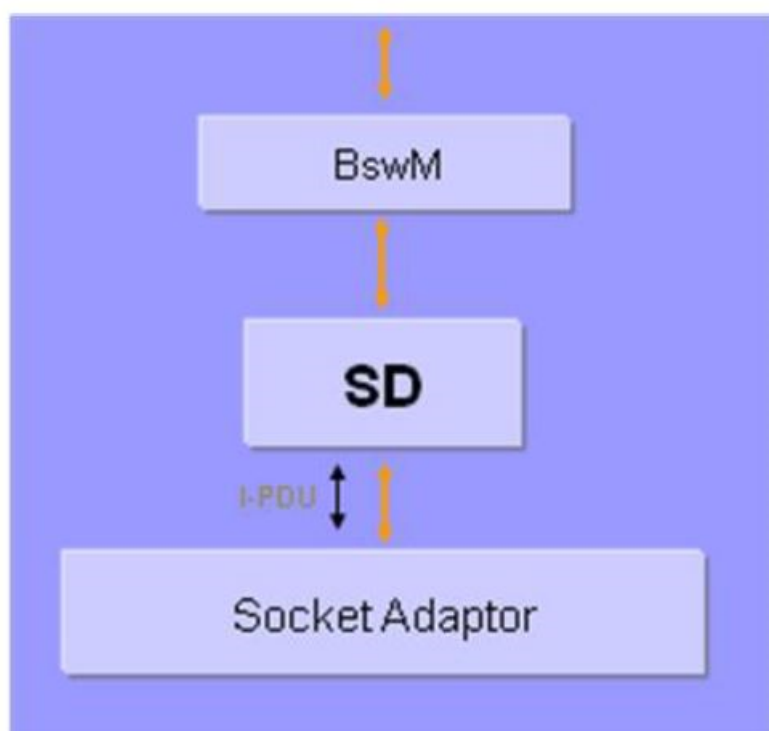


Figure 1.1: - Interaction of the AUTOSAR Service Discovery module

## 2 Acronyms, Abbreviations and Definitions

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

Abbreviation / Acronym:	Description:
BswM	Basis software manager
ECU	Electronic Control Unit
DEM	Diagnostic Event Manager
DET	Default Error Tracer
SD	Service Discovery
Sd	Service Discovery Module in AUTOSAR
SoAd	Socket Adaptor
SOME/IP	Scalable service-Oriented MiddlwarE over IP
SOME/IP-SD	SOME/IP Service Discovery

Term:	Description:
Service	A functional entity that offers an interface
Service Instance	A single instance of the Service
Offer	A message entry that offers a Service Instance
Stop Offer	A message that stops offering a Service Instance
Find	A message entry used to find a Service Instance
Event	A message sent by an ECU implementing a Service Instance to an ECU using this Service Instance.
Eventgroup	A logical grouping of 1 or more events. An Eventgroup is part of a Service.
Server Service	Provide a service
Client Service	Consumes a service
Server	A ECU which host ServerServices
Client	A ECU which host ClientServices
Endpoint Option	Endpoint Options are used to announce a tuple of unicast address and port
Multicast Option	Multicast Options are used to announce a tuple of multicast address and port
Unicast event	Events which are transmitted to a unicast endpoint by the ECU which host an SdServerService. The unicast endpoint is provided by a particular SdClientService which has subscribed to this SdServerService within the Endpoint Option referenced by a SubscribeEventgroup entry (see Consumed Eventgroup unicast endpoint)
Multicast event	Events which are transmitted to a multicast endpoint by the ECU which host an SdServerService. A multicast endpoint could be provided by the SdServerService (see Eventhandler multicast endpoint) and SdClientService (see Consumed Eventgroup multicast endpoint).





Term:	Description:
Eventhandler multicast endpoint	Term to describe the tuple of multicast address and port, which is pre-configured for a SdServerService per Eventhandler. If the threshold for subscribed Clients with different endpoint information has been reached, then the Server sends the corresponding events to this pre-configured multicast address and port. The Eventhandler multicast endpoint is announced via a <b>Multicast option</b> referenced by a SubscribeEventgroupAck entry
Consumed Eventgroup unicast endpoint	Term to describe the tuple of unicast address and port, which is pre-configured for a SdClientService per Consumed Eventgroup. A SdClientService which subscribes with this unicast address and port, indicates the SdServer to which endpoint, the corresponding events shall be sent. The Consumed Eventgroup unicast endpoint is announced via a <b>Endpoint option</b> referenced by a SubscribeEventgroup or StopSubscribeEventgroup entry
Consumed Eventgroup multicast endpoint	Term to describe the tuple of multicast address and port, which is pre-configured for a SdClientService per Consumed Eventgroup. A SdClientService which subscribe with this multicast address and port, indicates the SdServer to which endpoint the corresponding events shall be sent. The Consumed Eventgroup multicast endpoint is announced via a <b>Multicast option</b> referenced by a SubscribeEventgroup or StopSubscribeEventgroup
Eventhandler multicast connection	Term to describe the usage of an established socket connection if a SdServerService provides the Multicast events via the configured Eventhandler multicast endpoint
Consumed Eventgroup unicast connection	Term to describe the usage of an established socket connection if a SdClientService receives the events via a Consumed Eventgroup unicast endpoint
Consumed Eventgroup multicast connection	Term to describe the usage of an established socket connection if a SdClientService receives the events via a Consumed Eventgroup multicast endpoint

## 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] General Requirements on Basic Software Modules  
AUTOSAR\_CP\_SRS\_BSWGeneral
- [4] Layered Software Architecture  
AUTOSAR\_CP\_EXP\_LayeredSoftwareArchitecture
- [5] Requirements on SOME/IP Service Discovery Protocol  
AUTOSAR\_FO\_RS\_SOMEIPServiceDiscoveryProtocol
- [6] SOME/IP Service Discovery Protocol Specification  
AUTOSAR\_FO\_PRS\_SOMEIPServiceDiscoveryProtocol
- [7] Specification of Basic Software Mode Manager  
AUTOSAR\_CP\_SWS\_BSWModeManager
- [8] Specification of Socket Adaptor  
AUTOSAR\_CP\_SWS\_SocketAdaptor

### 3.2 Related specification

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

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

[2, SWS BSW General] [3, SRS General] [4, EXP Layered Software Architecture] [5, RS SOME/IP Service Discovery Protocol] [6, PRS SOME/IP Service Discovery Protocol] [7, SWS Basic Software Mode Manager] [8, SWS Socket Adaptor]

## 4 Constraints and assumptions

### 4.1 Limitations

Although the AUTOSAR SD is able to respond to wildcard requests (ANY) for Service ID, Instance ID, Major Version, and Minor Version, this module is only able to send wildcard finds for Minor Version.

This document does not yet contain trace links to the SRS Ethernet, therefore, the trace table is empty.

Load Balancing Option (Priority field and Weight field) can be configured for the Offer Services. However, the Client does not evaluate these fields.

The specification does not support setting the Discardable flag of a SOME/IP entry option to 1 and reacting to the reception of an (unknown/unsupported) option with the Discardable flag set to 1 (see [PRS\_SOMEIPSD\_00273], [PRS\_SOMEIPSD\_00275], [PRS\_SOMEIPSD\_00276], [PRS\_SOMEIPSD\_00544]).

The specification does not support that IPv4/IPv6 SD endpoint options are included in any SOME/IP-SD entry of a transmitted SOME/IP-SD message (see [PRS\_SOMEIPSD\_00547] - [PRS\_SOMEIPSD\_00552], [PRS\_SOMEIPSD\_00554] - [PRS\_SOMEIPSD\_00559], [PRS\_SOMEIPSD\_00650], [PRS\_SOMEIPSD\_00651], [PRS\_SOMEIPSD\_00654]).

SOME/IP-ACL contents will be the IP addresses of the allowed communication partners, so this feature will not be applicable in case of dynamic IP via DHCP is used.

### 4.2 Limitations for communication with Adaptive Platform

The following limitations regarding the SOME/IP SD functionality described in SOME/IP Service Discovery Protocol Specification and System Template apply:

- Configuration options (see [PRS\_SOMEIPSD\_00276] - [PRS\_SOMEIPSD\_00287])  
Capability records that are received from CP side will not be evaluated on AP side.

### 4.3 Applicability to car domains

N/A

## 5 Dependencies to other modules

### 5.1 AUTOSAR BSW Scheduler

The BSW Scheduler calls the main functions of the Service Discovery module, which is necessary for the cyclic processes of the Service Discovery.

### 5.2 AUTOSAR BSW Mode Manager

The BswM module provides the link between the generic mode requests and the service requests.

### 5.3 AUTOSAR Socked Adaptor

The Socked Adaptor hands over service requests between the Ethernet Stack and the Service Discovery Module.

The Service Discovery module shall be able to activate and de-activate the PDU routing from and to TCP/IP-sockets and trigger the initial transport of events (triggered transmit).

The SoAds Socket Connection Table needs to be pre-configured to receive the unicast and multicast messages sent by Service Discovery modules of other ECUs. As the ECU might be connected to multiple (virtual) networks, there can exist multiple Service Discovery Instances, which may have multiple Socket Connection Table entries. The triples of Unicast Rx, Multicast Rx, and Tx PduIDs for each (virtual) interface need to be configured in the SoAd and known to the Service Discovery module.

Additionally the Service Discovery module updates endpoint information (IP address and port number) in socket connections (SoAdSocketConnection), which the Service Discovery module extracts from received Service Discovery messages.

For robustness reasons these UDP Sockets should only be used for SD messages and the option `SoAdSocketUdpStrictHeaderLenCheckEnabled` should be turned on.

### 5.4 AUTOSAR Default Error Tracer

In order to be able to report development errors, the Service Discovery module has to have access to the error hook of the Default Error Tracer.

## 5.5 AUTOSAR Diagnostic Event Manager

In order to be able to report production errors the Service Discovery module has to have access to the Diagnostic Event Manager.

## 5.6 AUTOSAR Non Volatile Memory

In order to save the updated ACL (Access Control List), the Service Discovery module shall have access to Non Volatile Memory NVM.

## 5.7 File structure

### 5.7.1 Code file structure

**[SWS\_Sd\_00001]** [The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- Sd\_Lcfg.c - for link time configurable parameters and
- Sd\_PBcfg.c - for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.]()

### 5.7.2 Header file structure

**[SWS\_Sd\_00003]** [The module shall include the Dem.h file. By this inclusion, the APIs to report errors as well as the required Event Id symbols are included.]()

**[SWS\_Sd\_00762]{DRAFT}** [The module shall include the header file NvM.h if the ACL check is configured.]()



## 6 Requirements Tracing

The following tables reference the requirements specified in [5] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[RS_Ids_00810]	Basic SW security events	<a href="#">[SWS_Sd_00114]</a>
[RS_SOMEIPSD_-00025]	SOME/IP Service Discovery messages shall contain information how to contact the communication partner	<a href="#">[SWS_Sd_00799]</a> <a href="#">[SWS_Sd_00800]</a>
[SRS_Eth_00161]	Service Provider Check	<a href="#">[SWS_Sd_00785]</a> <a href="#">[SWS_Sd_00786]</a> <a href="#">[SWS_Sd_00787]</a> <a href="#">[SWS_Sd_00788]</a>
[SRS_Eth_00162]	Event Subscriber Check	<a href="#">[SWS_Sd_00789]</a> <a href="#">[SWS_Sd_00790]</a> <a href="#">[SWS_Sd_00791]</a>
[SRS_Eth_00163]	Method Call Request Check	<a href="#">[SWS_Sd_00792]</a> <a href="#">[SWS_Sd_00793]</a> <a href="#">[SWS_Sd_00794]</a> <a href="#">[SWS_Sd_00795]</a> <a href="#">[SWS_Sd_00796]</a> <a href="#">[SWS_Sd_91011]</a>
[SRS_Eth_00164]	ACL Policy Configuration	<a href="#">[SWS_Sd_00763]</a> <a href="#">[SWS_Sd_00764]</a> <a href="#">[SWS_Sd_00765]</a> <a href="#">[SWS_Sd_00766]</a> <a href="#">[SWS_Sd_00767]</a>
[SRS_Eth_00165]	ACL Update	<a href="#">[SWS_Sd_00768]</a> <a href="#">[SWS_Sd_00769]</a> <a href="#">[SWS_Sd_00780]</a> <a href="#">[SWS_Sd_00781]</a> <a href="#">[SWS_Sd_00782]</a> <a href="#">[SWS_Sd_00783]</a> <a href="#">[SWS_Sd_00784]</a> <a href="#">[SWS_Sd_91009]</a> <a href="#">[SWS_Sd_91010]</a>
[SRS_Eth_00166]	Security Alerts Raising	<a href="#">[SWS_Sd_00797]</a>

**Table 6.1: RequirementsTracing**

## 7 Functional specification

### 7.1 Background & Rationale

The main tasks of the Service Discovery Module are managing the availability of functional entities called services in the in-vehicle communication as well as controlling the send behavior of event messages. This allows sending only event messages to receivers requiring them (Publish/Subscribe). The solution described here is also known as SOME/IP-SD (Scalable service-Oriented MiddlewarE over IP - Service Discovery).

With Service Discovery different ECUs can offer Service Instances and find available Service Instances within the vehicle network. An ECU can stop offering a Service Instance it was offering before. Later finds to such a service instance will remain unanswered. Service Instances are single implementations of a service that is defined by its service interface. In the AUTOSAR context, a find is an operation to identify available Service Instances and their locations.

In addition to the status of Service Instances, the Service Discovery is able to control sending special messages called events. These events are grouped into Eventgroups, which the Service Discovery can turn on/off in a Publish/Subscribe manner; thus, turning the sending and receiving of the events of this Eventgroup on/off.

For the remainder of this document, the definitions listed in Chapter 2 apply.

Figure 7.1 shows the interaction between Services and Eventgroups. On the abstract level, the service can contain zero to many Eventgroups. However, when creating the overall system, this information has to be configured into different ECUs with different roles (clients and servers). When instantiating the Services and the contained Eventgroups, the ServerServices and ClientServices as well as the EventHandlers and ConsumedEventgroups are instantiated from the Services and Eventgroups.

A local ECU needs to deal with two different kinds of services:

- Server Services - The local ECU **offers** Server Service Instances (i.e. located locally) to the rest of the vehicle and can be considered the server for this Service Instance.
- Client Services - The local ECU **may use** Server Service Instances offered by another ECU inside the vehicle and can be considered a client to this Service Instance.

For Server Services the local ECUs Service Discovery module has to (server role):

- Offer the local service, when it is available; i.e. the SWC(s) offering the service are ready and the service is available in the current state of the ECU.
- Take back the offer of the local service (stop offer), when the service is no longer available.
- Answer and respond to finds of other ECUs.

For Client Services the local ECUs Service Discovery module has to (client role):

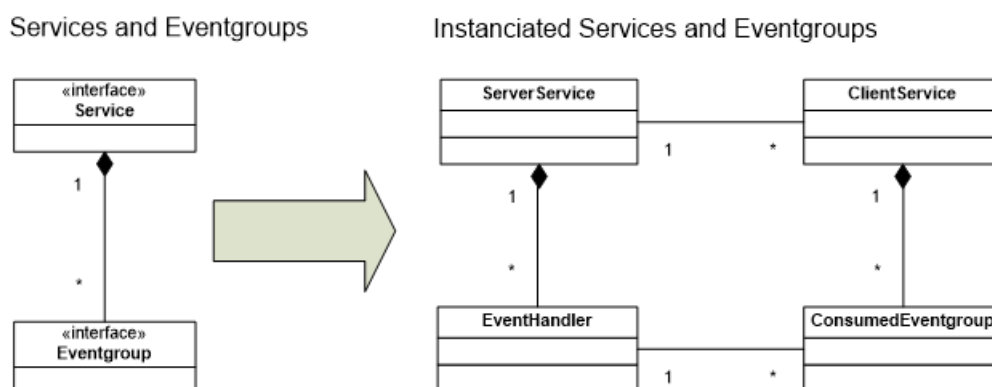
- Listen for offers and finds depending of the configuration store this information in volatile memory.
- Listen for stop offers and depending of the configuration store this information in volatile memory.
- Send finds depending on the state of the current ECU and its SWCs.

Service Discovery can be used to manage Publish/Subscribe relationships as well. In the Service Discovery based Publish/Subscribe use-case one ECU (Publish/Subscribe Client with ConsumedEventgroup) is interested in receiving some data from (subscribing to) another ECU (Publish/Subscribe Server with EventHandler).

While the Subscribe is defined explicitly in the SD message, the Publish is based on the availability of the service Instance itself (OfferService entry). Based on the offered Service Instance the Publish/Subscribe Client may subscribe via SubscribeEventgroup entries. The Publish/Subscribe Server will now use this subscription to register the Publish/Subscribe Client as an interested party in some information specified by the subscription and start sending that information to the Publish/Subscribe Client pending some event or time-out.

As optimization, the SD supports sending event messages to multiple clients using multicast messages instead of a unicast message per client. Please note, it has to be differed between a multicast endpoint which could be pre-configured on Server side and a multicast endpoint which could be pre-configured on Client side:

1. If an SdServerService has a pre-configured multicast address and port per EventHandler, then the SdServerService switches to this multicast address and port (so-called "EventHandler multicast endpoint"), if the threshold (SdEventHandler-MulticastThreshold) for subscribed SdClientServices with different endpoint information has been reached
2. If an SdClientService has subscribed with a multicast address and port (so-called "Consumed Eventgroup multicast endpoint"), then the SdServerService sends its events upon a subscription to the Consumed Eventgroup multicast endpoint (multicast address and port)



**Figure 7.1: - Overview of Services and Eventgroups**

## 7.2 Requirements

### 7.2.1 General requirements

**[SWS\_Sd\_00400]** [It shall be possible to configure the Service Discovery module as an optional AUTOSAR BSW Module. Please refer to the SystemTemplate for configuration.]()

**[SWS\_Sd\_00004]** [The Service Discovery shall implement a main function, which shall be called cyclically according to configuration parameter SdMainFunctionCycleTime.]()

**[SWS\_Sd\_00005]** [The Service Discovery module shall store the ServiceModeRequest, which is provided via the BswM by calling the following APIs:

- Sd\_ServerServiceSetState() and Sd\_ClientServiceSetState(), respectively, If the SdServerService and SdClientService, respectively, is NOT referencing a SdServiceGroup
- Sd\_ServiceGroupStart and SdServiceGroupStop, if the SdServerService and Sd ClientService, respectively is referencing a SdServiceGroup
- Sd\_ConsumedEventGroupSetState(), if dedicated SdEventGroupS are requested by a SdClientService. (**Note:** This API call is allowed independ of a reference to a SdServiceGroup of a SdClientService )
- Sd\_EventHandlerSetState() does currently not exist, since this state is directly deduced from the state of a Server Service by the Service Discovery.

]()

**Note:**

Based on the interaction with SWCs, the following modes can be requested by the Bsw M module:

Server SWCs via Sd\_ServerServiceSetState() or, Sd\_ServiceGroupStart() and Sd\_ServiceGroupStop(), respectively:

- SD\_SERVER\_SERVICE\_DOWN
- SD\_SERVER\_SERVICE\_AVAILABLE

Client SWCs via Sd\_ClientServiceSetState() or, Sd\_ServiceGroupStart() and Sd\_ServiceGroupStop(), respectively:

- SD\_CLIENT\_SERVICE\_RELEASED
- SD\_CLIENT\_SERVICE\_REQUESTED

Client SWCs via Sd\_ConsumedEventGroupSetState()

- SD\_CONSUMED\_EVENTGROUP\_RELEASED

- SD\_CONSUMED\_EVENTGROUP\_REQUESTED

"SD\_SERVER\_SERVICE\_DOWN" implies that the local SWC(s) offering this Service Instance are not ready to communicate,

"SD\_SERVER\_SERVICE\_AVAILABLE" implies that the local SWC(s) offering this Service Instance are ready to communicate,

"SD\_CLIENT\_SERVICE\_RELEASED" implies that the local SWC(s) using this Service Instance do not need to communicate with this Service Instance,

"SD\_CLIENT\_SERVICE\_REQUESTED" implies that the local SWC(s) using this service is ready to communicate with this Service Instance and needs this Service Instance,

"SD\_CONSUMED\_EVENTGROUP\_RELEASED" implies that the local SWC(s) using this Consumed Eventgroup do not need the events of this Consumed Eventgroup,

"SD\_CONSUMED\_EVENTGROUP\_REQUESTED" implies that the local SWC(s) using this Consumed Eventgroup need the events of this Consumed Eventgroup.

**[SWS\_Sd\_00007]** [The following CurrentStates shall be available for reporting to BswM module via BswM\_Sd\_ClientServiceCurrentState(), BswM\_Sd\_ConsumedEventGroupCurrentState(), and BswM\_Sd\_EventHandlerCurrentState() respectively:

- SD\_CLIENT\_SERVICE\_DOWN
- SD\_CLIENT\_SERVICE\_AVAILABLE
- SD\_CONSUMED\_EVENTGROUP\_DOWN
- SD\_CONSUMED\_EVENTGROUP\_AVAILABLE
- SD\_EVENT\_HANDLER\_RELEASED
- SD\_EVENT\_HANDLER\_REQUESTED]

**Note:**

"SD\_CLIENT\_SERVICE\_DOWN" tells the local SWC(s) that this Service Instance is not available,

"SD\_CLIENT\_SERVICE\_AVAILABLE" tells the local SWC(s) that this Service Instance is available,

"SD\_CONSUMED\_EVENTGROUP\_DOWN" tells the local SWC(s) that this Consumed Eventgroup is not currently subscribed,

"SD\_CONSUMED\_EVENTGROUP\_AVAILABLE" tells the local SWC(s) that this Consumed Eventgroup is currently subscribed (i.e. events are received),

"SD\_EVENT\_HANDLER\_RELEASED" tells the local SWC(s) that no client is currently subscribed to this Eventgroup,

"SD\_EVENT\_HANDLER\_REQUESTED" tells the local SWC(s) that at least one client is currently subscribed to this Eventgroup.

**[SWS\_Sd\_00011]** [Every configured Server Service Instance shall have an ECU wide, unique SdServerServiceHandleId.] ()

**[SWS\_Sd\_00437]** [Every configured Client Service Instance shall have an ECU wide, unique SdClientServiceHandleId.] ()

**[SWS\_Sd\_00438]** [Every configured Consumed Event Group shall have an ECU wide, unique SdConsumedEventGroupHandleId.] ()

**[SWS\_Sd\_00439]** [Every configured Event Handler shall have an ECU wide, unique SdEventHandlerHandleId.] ()

**Note for [SWS\_Sd\_00011], \_00437, \_00438, and \_00439:**

The IDs defined by the above requirements are needed in order to identify the Service Instances and Eventgroups in the control API between Sd and BswM.

This is even valid for Instances or Eventgroups with the same Service ID and/or the same Service Instance ID.

## 7.2.2 Ethernet Communication

**[SWS\_Sd\_00013]** [Every Service Discovery Configuration Instance (see configuration container SdInstance) shall have at least one TxPdu ID, one RxPdu ID for Unicast, and one RxPdu ID for Multicast (see configuration parameter SdInstanceTxPdu, SdInstanceUnicastRxPdu, and SdInstanceMulticastRxPdu respectively).] ()

**[SWS\_Sd\_00017]** [For different links, separate Service Discovery instance containers shall be configured.] ()

### Note:

Links in this regards also includes different virtual links using Ethernet VLANs.

**[SWS\_Sd\_00697]** [A SD Instance does only support a single Address Family (i.e. IPv4 or IPv6). This address family shall be learned by means of the SoAd configuration of SdInstanceTxPdu, SdInstanceUnicastRxPdu, and SdInstanceMulticastRxPdu (local address).] ()

**[SWS\_Sd\_00723]** [During initialization of the SD module, the API SoAd\_OpenSoCon() shall be called for all Socket Connections associated with SdInstanceTxPdu, SdInstanceUnicastRxPdu and SdInstanceMulticastRxPdu.] ()

### Note:

The SoAd module needs to be initialized before the SD module is initialized.

### Note:

An implementer has to guarantee that `SoAd_SetUniqueRemoteAddr()`, `SoAd_GetLocalAddr()`, and `SoAd_SetRemoteAddr()` can never return errors by validating the source code and configuration of Service Discovery and Socket Adaptor. Failures of `SoAd_SetUniqueRemoteAddr()`, `SoAd_GetLocalAddr()`, and `SoAd_SetRemoteAddr()` cannot be recovered from.

### 7.2.3 State Handling

**[SWS\_Sd\_00019]** [The Service Discovery module shall store the status of all statically configured Service Instances and Eventgroups separately.]()

**[SWS\_Sd\_00020]** [After initialization of the Service Discovery module by the call of the API `Sd_Init()`, all configured Server Service Instances shall have the state "SD\_SERVER\_SERVICE\_DOWN", unless a Server Service Instance has `SdServerServiceAutoAvailable` set to true, then the state shall be set to "SD\_SERVER\_SERVICE\_AVAILABLE".]()

**Note:**

`SdServerServiceAutoAvailable` set to true, is only allowed for Server Services which are NOT referencing a `SdServiceGroup`.

**[SWS\_Sd\_00021]** [After initialization of the Service Discovery module by calling of the API `Sd_Init()`, all configured Client Service Instances shall have the state "SD\_CLIENT\_SERVICE\_RELEASED", unless a Client Service Instance has `SdClientServiceAutoRequired` set to true, then the state shall be set to "SD\_CLIENT\_SERVICE\_REQUESTED".]()

**Note:**

`SdClientServiceAutoRequire` set to true, is only allowed for Client Services which are NOT referencing a `SdServiceGroup`.

**[SWS\_Sd\_00440]** [After initialization of the Service Discovery module by calling of the API `Sd_Init()`, all configured Eventgroups shall have the state "SD\_CONSUMED\_EVENTGROUP\_RELEASED", unless a Consumed Eventgroup has "`SdConsumedEventGroupAutoRequired`" set to true, then the state shall be set to "SD\_CONSUMED\_EVENTGROUP\_REQUESTED" as soon as the associated Client Service Instance is requested.]()

**[SWS\_Sd\_00402]** [The Service Discovery module shall store all IP address assignment states referenced by server and client Service Instances.]()

**[SWS\_Sd\_00442]** [If `Sd_ConsumedEventGroupSetState` is called with `SD_CONSUMED_EVENTGROUP_REQUESTED` while its Client Service Instance is still released (`SD_CLIENT_SERVICE_RELEASED`) `E_NO_OK` shall be returned.]()

**[SWS\_Sd\_00443]** [If a `SdClientService` is set to `SD_CLIENT_SERVICE_RELEASED` (via `Sd_ClientServiceSetState()` or `Sd_ServiceGroupStop()`) while one or more of



its Eventgroups are still requested (SD\_CONSUMED\_EVENTGROUP\_REQUESTED) the Service Discovery shall interpret this the same way as these Eventgroups were called with SD\_CONSUMED\_EVENTGROUP\_RELEASED first. ]()

#### 7.2.4 Interaction with Socket Adaptor

**[SWS\_Sd\_00024]** [The Service Discovery module shall be able to enable/disable routing groups within the SoAd module using the APIs SoAd\_EnableRouting(), SoAd\_DisableRouting(), SoAd\_EnableSpecificRouting(), and SoAd\_DisableSpecificRouting() for Server- and Client Service Instances.]()

**[SWS\_Sd\_00699]** [The Service Discovery module shall be able to trigger the sending of initial Events using the API SoAd\_IfSpecificRoutingGroupTransmit().]()

**[SWS\_Sd\_00026]** [The Service Discovery module shall be able to reference Routing Group(s) per Service Instance/Eventgroup. See the following configuration parameters:

- SdClientServiceActivationRef (in SdConsumedMethods)
- SdConsumedEventGroupMulticastActivationRef
- SdConsumedEventGroupTcpActivationRef
- SdConsumedEventGroupUdpActivationRef
- SdServerServiceActivationRef (in SdProvidedMethods)
- SdEventActivationRef (in SdEventHandlerMulticast)
- SdEventActivationRef (in SdEventHandlerTcp)
- SdEventTriggeringRef (in SdEventHandlerTcp)
- SdEventActivationRef (in SdEventHandlerUdp)
- SdEventTriggeringRef (in SdEventHandlerUdp)

]()

**[SWS\_Sd\_00700]** [The Service Discovery module shall be able to reference Socket Connections and SocketConnectionGroups per Service Instance/Eventgroup. See the following configuration parameters:

- SdClientServiceTcpRef (Service Instance and Eventgroups)
- SdClientServiceUdpRef (Service Instance and Eventgroups)
- SdConsumedEventGroupMulticastGroupRef (Eventgroup)
- SdServerServiceTcpRef (Service Instance and Eventgroups)
- SdServerServiceUdpRef (Service Instance and Eventgroups)



- SdMulticastEventSoConRef in SdEventHandlerMulticast (Eventgroup)

]()

**[SWS\_Sd\_00029]** [The Service Discovery module shall only call SoAd\_IfTransmit() if an IP address is assigned; i.e.: Sd\_LocalIpAddrAssignmentChg() has been called with the current state TCPIP\_IPADDR\_STATE\_ASSIGNED.]()

**[SWS\_Sd\_00709]** [Ignore, if SoAd\_IfTransmit() returns E\_NOT\_OK.]()

**[SWS\_Sd\_00481]** [Every wildcard socket connection shall be reset to wildcard using SoAd\_ReleaseRemoteAddr() if all of the following conditions apply:

- The remote address of a socket connection has been set by SD.
- The socket connection is not used by a ClientService anymore. I.e. no Offer was received, a Stop Offer was received or the TTL has expired.
- The socket connection is not used by an Eventhandler anymore. I.e. the client has unsubscribed all Eventgroups using this socket connection. The socket connection shall not be reset if the routings get disabled because the SdEventHandlerMulticastThreshold was reached.

]()

**Note:** This requirement does not apply to the socket connections used for service discovery.

## 7.2.5 Subscribe Eventgroup retry handling

The Subscribe Eventgroup retry mechanism is an optional feature for ClientServices. This could be used to speed up the recovery if a SOME/IP-SD message is lost (e.g. SubscribeEventGroupAck) and the interval between cycle offers are too large to get a fast recovery, or to speed up subscriptions if an Eventgroup is requested somewhere between two cyclic offers. The timing behavior of Subscribe Eventgroup retry mechanism could be configured per ClientService and has to match to the timing behavior of the corresponding ServerService (see TPS SysT constr\_5095). For ServerServices which have their TLL (SdServerTimerTTL) set to 0xFFFFFFFF and their interval between cyclic offers in the main phase (SdServerTimerOfferCyclicDelay) set to 0, it's possible to set the Subscribe Eventgroup retry to 0xFF (see TPS SysT constr\_5096). This would mean to retry the subscription to an EventGroup as long as the EventGroup is set to SD\_CONSUMED\_EVENTGROUP\_REQUESTED and no SubscribeEventGroup Ack was received.

**[SWS\_Sd\_00735]** [The subscribe Eventgroup retry handling shall only be processed for Eventgroups of a ServerService where

- SdSubscribeEventgroupRetryMax is greater than 0,
- and only if SdSubscribeEventgroupRetryEnable is set to TRUE.

]()

**[SWS\_Sd\_00736]** [If SdSubscribeEventgroupRetryEnable is set to TRUE and SdSubscribeEventgroupRetryMax is set to a value greater than 0, every time a Consumed Eventgroup transit to the state SD\_CONSUMED\_EVENTGROUP\_REQUESTED, the following actions shall be done:

- the corresponding client service subscription retry delay timer shall be started and set to SdSubscribeEventgroupRetryDelay, if the timer is not already running
- the Eventgroup subscription retry counter shall be initialized with 1

]()

**[SWS\_Sd\_00737]** [If the client service subscription retry delay timer elapsed and the counts of retries of subscription (SdSubscribeEventgroupRetryMax) did not exceed for a configured Eventgroup, client service subscription retry delay timer shall be re-initialized and the subscription for the Eventgroup shall be re-triggered by sending a combination of StopSubscribeEventgroup/SubscribeEventgroup, and the retry counter shall be incremented. If the counts of retries of subscription (SdSubscribeEventgroupRetryMax) exceeds, the ServiceDiscovery module shall raise the runtime error "SD\_E\_COUNT\_OF\_RETRY\_SUBSCRIPTION\_EXCEEDED".]()

**[SWS\_Sd\_00738]** [The retry of a subscription for a requested Eventgroup shall be stopped for the following conditions:

- If a SubscribeEventGroupAck or SubscribeEventGroupNack was received for the requested Eventgroup.
- If the count of retries exceeds SdEventgroupSubscribeRetryMax of the requested Eventgroup.
- If the requested Eventgroup is set to "SD\_CONSUMED\_EVENTGROUP\_RELEASED".

]()

**[SWS\_Sd\_00739]** [If SdSubscribeEventgroupRetryEnable is set to TRUE and SubscribeEventgroupRetryMax is set to 0xFF, the retries of subscription shall continue as long as all of the following conditions are fulfilled:

- the corresponding Eventgroup is set to "SD\_CONSUMED\_EVENTGROUP\_REQUESTED"
- no SubscribeEventGroupAck or no SubscribeEventGroupNack was received

]()

**[SWS\_Sd\_00740]** [The client service subscription retry delay timer shall be cancelled, if the retry is finished for all Eventgroups of a ClientService according to [\[SWS\\_Sd\\_00738\]](#).]()

When the client does not receive initial events before the next OfferService is received, it should stop requesting the eventgroup, i.e. trigger StopSubscribeEventgroup, and resume requesting the eventgroup, i.e. trigger SubscribeEventgroup when the next OfferService is received.

This procedure can be triggered on application level and corresponds functionally to a StopSubscribeEventgroup/SubscribeEventgroup combination after a loss of a SubscribeEventgroupAck. This might imply notifying the SD-Module about reception of the Initial Event of each and every Field, or other appropriate means.

If the procedures, described in the previous paragraphs cannot be implemented by the application, the retry-mechanism should be out-sourced to the BswM in a rule that initiates re-sending of Initial Events via triggering a StopSubscribeEventgroup/-SubscribeEventgroup SD message upon detecting that a security association is established, to increase at least the robustness for a security association based communication.

Since the set-up of an security association is asynchronous, the BswM rule (BswM-ModeRequestSource/BswMTimer) should thereby delay sending StopSubscribeEventgroup/SubscribeEventgroup by an appropriate time that allows both peers to finish establishing the security association.

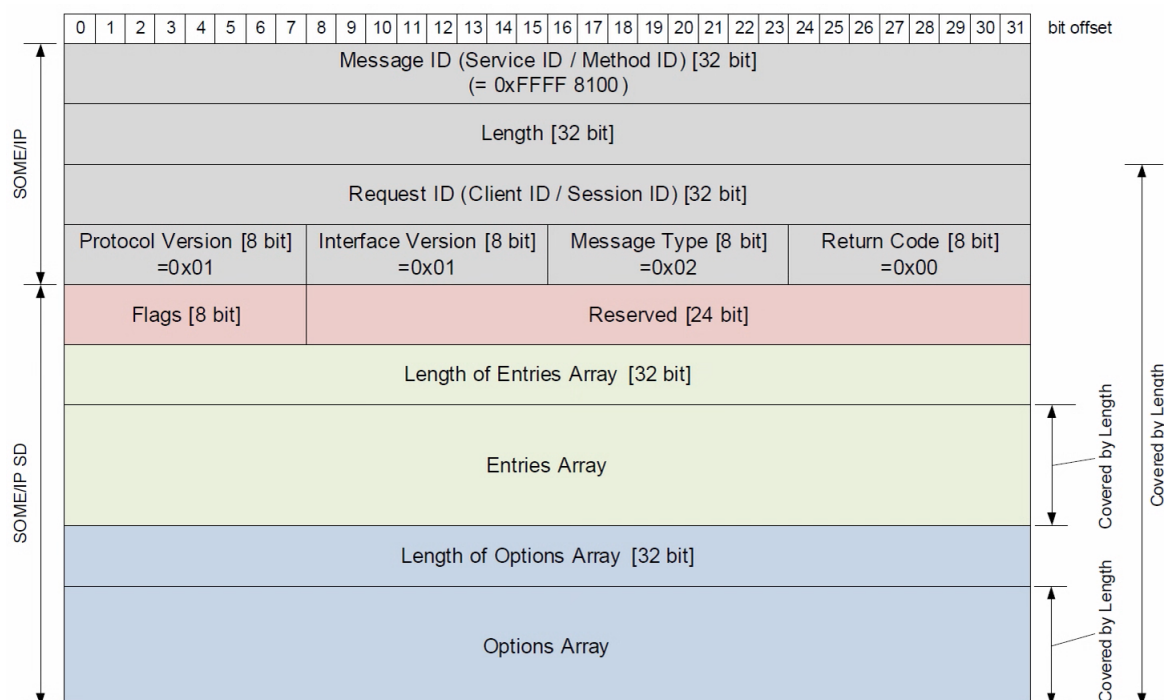
If the Subscribe Eventgroup Ack entry does not arrive before the next Subscribe Eventgroup entry is sent (see [PRS\_SOMEIPSD\_00463]) or if the client does not receive initial events before the next OfferService is received, this should not lead to re-establishing security association connections, if the current connection is being set-up or is already set-up.

For events that are transported using a security association the client has to make sure that the security association is established and that it is ready to receive messages before sending the SubscribeEventgroup entry (see [SWS\_Sd\_00761]). The server, on the other hand, has to make sure that the security association is established and that it is able to send messages before sending the SubscribeEventgroupAck entry (see [SWS\_Sd\_00760]).

**[SWS\_Sd\_00759]** [If a SubscribeEventgroup entry is received, for which a security association is required, and the security association not yet established, this entry shall be answered with a SubscribeEventgroupNack entry (see [SWS\_Sd\_00760]).] ()

## 7.3 Message format

Most of the requirements are handled in the PRS [5.1.2 SomeIP SD message format] and only the Classic Plattform related is defined in this chapter.



**Figure 7.2: - Overview of the Service Discovery message format**

**Note:** All Service Discovery messages are in Network Byte Order (i.e. Big Endian Byte Order), see [PRS\_SOMEIPSD\_00853]. This is ensured for SOME/IP header fields by [PRS\_SOMEIP\_00368] and needs to be configured for SOME/IP Payload according to [PRS\_SOMEIP\_00369].

### 7.3.1 Request ID

This chapter describes the requirements related to the Request ID field. The Request ID is made up of Client ID and Session ID. While the Client ID is not used for Service Discovery, the Session ID is used to detect the reboot or restart of other Service Discovery instances in the vehicle in order to repair the local state of the Service Discovery module.

**[SWS\_Sd\_00034]** [After initialization of the Service Discovery Module, the Session ID for messages sent by the local ECU shall be 0x0001.]()

**Note to [SWS\_Sd\_00034]:** This means that the first SD message sent out has Session ID set to 0x0001. According to [PRS\_SOMEIPSD\_00160] the Service Discovery module has to handle the session ID per communication partner. Thus, the first SD message sent out to the multicast endpoint as well as the first SD message sent out to any unicast endpoint has the Session ID set to 0x0001.

### 7.3.2 Protocol Version field

The Protocol Version field is used to describe the current version of SOME/IP.

### 7.3.3 Interface Version field

The Interface Version field is used to describe the current version of the SOME/IP service; i.e. the current version of SOME/IP-SD itself.

### 7.3.4 Message Type field

The Message Type field is used to differentiate the types of SOME/IP messages. SOME/IP-SD uses only event messages; thus, it always uses the same type.

### 7.3.5 Return Code field

The Return Code is used to signal whether a request was successfully been processed. This is not applicable for SOME/IP-SD; therefore, the return code will be statically set to 0x00.

### 7.3.6 Flags field

With the Flags field the SOME/IP-SD header starts. It is used to signal global Service Discovery information, which includes currently the state of the last reboot as well as the capability of receiving unicast messages.

**[SWS\_Sd\_00448]** [If a server or client detects a reboot by evaluating the Session ID and Reboot Flag of a received SOME/IP-SD message which was send by a communication partner, then the local state of the affected communication partner shall expire and the following actions shall be performed:

- In case a client detects a reboot of a server and the client uses a service of this server, the client shall handle the reboot as if a StopOffer entry was received (see also [\[SWS\\_Sd\\_00367\]](#) for further details). Furthermore
  - If SdClientServiceTcpRef is configured for this service, the active Client shall close the corresponding TCP connection by calling SoAd\_CloseSoCon() with parameter "abort" set to TRUE.
  - This Offer entry shall be processed according to [\[SWS\\_Sd\\_00721\]](#).
  - In case a server detects a reboot of a client and the client uses a service of this server, the server shall handle the reboot as if a StopSubscribeEvent-

group entry was received (see also [SWS\_Sd\_00345] for further details). Furthermore

- \* If SdServerServiceTcpRef is configured for this service, the active Server shall close the corresponding TCP connection by calling SoAd\_CloseSoCon() with parameter "abort" set to TRUE , and re-establish the TCP connection again by calling SoAd\_OpenSoCon().
- \* Afterwards this message shall be processed according to [SWS\_Sd\_00343] or [SWS\_Sd\_00344].

]()

**Note:** A call of SoAd\_CloseSoCon() with parameter "abort" set to TRUE will terminate immediately the TcpIp connection by sending an TCP package with reset flag (RST flag) set.

### 7.3.7 Reserved field

This Reserved field is not currently used and left empty for further enhancements of the SOME/IP-SD protocol.

### 7.3.8 Entries Array

When SOME/IP-SD find or offers Service Instances or handles subscriptions this is done by so called entries, which are transported in the entry array of the SOME/IP-SD message (see Figure 7.2).

#### 7.3.8.1 Entry Format Type 1

Two types of Entries exist: Type 1 Entries for Services and Type 2 Entries for Event-groups.

For further details on the Entry Format Type 1 , see [6, PRS SOME/IP Service Discovery Protocol] Chapter 4.1.2.3 Entry Format

The Type 1 Entries shall have the following layout:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	bit offset
Type								Index 1st options								Index 2nd options								# of opt 1				# of opt 2				
Service ID																Instance ID																
Major Version								TTL																								
Minor Version																																

**Figure 7.3: - Layout of Type 1 Entries (Entries for Services)**

**[SWS\_Sd\_00173]** [The Service ID field of the Type 1 Entry format layout shall carry the Service ID of the service, statically configured using the parameter SdServerServiceID and SdClientServiceID, depending on being a server or client entry. See also [\[\[ECUC\\_SD\\_00009\]\]](#) and [\[\[ECUC\\_SD\\_00020\]\]](#).] ()

**[SWS\_Sd\_00175]** [The Instance ID field of the Type 1 Entry format layout shall carry the Instance ID of the service, statically configured using the parameter SdServerServiceInstanceID and SdClientServiceInstanceID, depending on being a server or client entry. See also [\[\[ECUC\\_SD\\_00011\]\]](#) and [\[\[ECUC\\_SD\\_00022\]\]](#).] ()

**[SWS\_Sd\_00178]** [The Major Version field of the Type 1 Entry format layout shall carry the SdServerServiceMajorVersion and SdClientServiceMajorVersion, depending on being a server or client entry. See also [\[\[ECUC\\_SD\\_00068\]\]](#) and [\[\[ECUC\\_SD\\_00070\]\]](#).] ()

**[SWS\_Sd\_00180]** [The TTL field of the Type 1 Entry format layout defines the lifetime of the entry for Servers in seconds configured using the parameter SdServerTimerTTL and SdClientTimerTTL, except for Stop-Entries, which have a TTL of 0. See also [\[\[ECUC\\_SD\\_00037\]\]](#) and [\[\[ECUC\\_SD\\_00075\]\]](#).]

**Note:** For Clients the TTL value is not used for Type 1 Entries and shall be ignored by the server service.] ()

**[SWS\_Sd\_00182]** [The Minor Version field of the Type 1 Entry format layout shall carry the SdServerServiceMinorVersion and SdClientServiceMinorVersion. See also [\[\[ECUC\\_SD\\_00069\]\]](#) and [\[\[ECUC\\_SD\\_00071\]\]](#).] ()

### 7.3.8.2 Entry Format Type 2

The Type 2 Entries format shall be used for Eventgroups.

For further details on the Entry Format Type 2 , see [\[6, PRS SOME/IP Service Discovery Protocol\]](#) Chapter 4.1.2.3 Entry Format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	bit offset
Type								Index 1st options								Index 2nd options								# of opt 1				# of opt 2				
Service ID																Instance ID																
Major Version								TTL																								
Reserved (0x000)											Counter					Eventgroup ID																

**Figure 7.4: - Layout of Type 2 Entries (Entries for Eventgroups)**

**[SWS\_Sd\_00193]** [The Service ID field of the Type 2 Entry format layout shall carry the Service ID of the eventgroups service, statically configured using the parameter SdServerServiceID and SdClientServiceID, depending on being a server or client entry. See also [\[\[ECUC\\_SD\\_00009\]\]](#) and [\[\[ECUC\\_SD\\_00020\]\]](#).] ()



**[SWS\_Sd\_00195]** [The Instance ID field of the Type 2 Entry format layout shall carry the Instance ID of the eventgroups service statically configured using the parameter SdServerServiceInstanceID and SdClientServiceInstanceID, depending on being a server or client entry. See also [\[\[ECUC\\_SD\\_00011\]\]](#) and [\[\[ECUC\\_SD\\_00022\]\]](#).]()

**[SWS\_Sd\_00198]** [The Major Version field of the Type 2 Entry format layout shall carry the SdServerServiceMajorVersion and SdClientServiceMajorVersion, depending on being a server or client entry. See also [\[\[ECUC\\_SD\\_00068\]\]](#) and [\[\[ECUC\\_SD\\_00070\]\]](#).]()

**[SWS\_Sd\_00200]** [The TTL field of the Type 2 Entry format layout defines the lifetime of the entry in seconds configured using the parameter SdServerTimerTTL and SdClientTimerTTL, except for Stop- or Nack-Entries, which use a TTL of 0. See also [\[\[ECUC\\_SD\\_00037\]\]](#) and [\[\[ECUC\\_SD\\_00075\]\]](#).]()

**[SWS\_Sd\_00204]** [The Eventgroup ID field of the Type 2 Entry format layout shall carry the ID of an Eventgroup, configured using the parameter SdConsumedEventGroupID. See also [\[\[ECUC\\_SD\\_00057\]\]](#).]()

**[SWS\_Sd\_00476]** [Type 2 Entries (Entries for Eventgroups) shall not use "any values" as Service ID (i.e. 0xFFFF), Instance ID (i.e. 0xFFFF), Eventgroup ID (i.e. 0xFFFF), and/or Major Version (i.e. 0xFF).]()

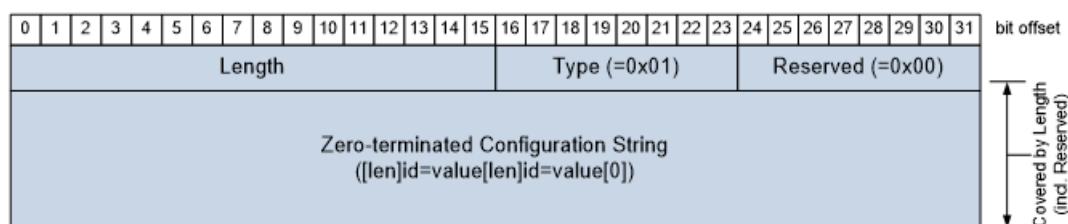
### 7.3.9 Options Array

The Option array is the last part of the Service Discovery Message (see [Figure 7.2](#)). The options in the options array carry additional information.

For further details on the Configuration Option, see [\[6, PRS SOME/IP Service Discovery Protocol\]](#) Chapter 4.1.2.4 Options Format

#### 7.3.9.1 Configuration Option

The Configuration Option transports additional attributes of entries in the Service Discovery messages. Between 0 and n configuration items can be transported using the Configuration Option. These configuration items can include for example the name of the host or the Service.



**Figure 7.5: - Configuration Option**

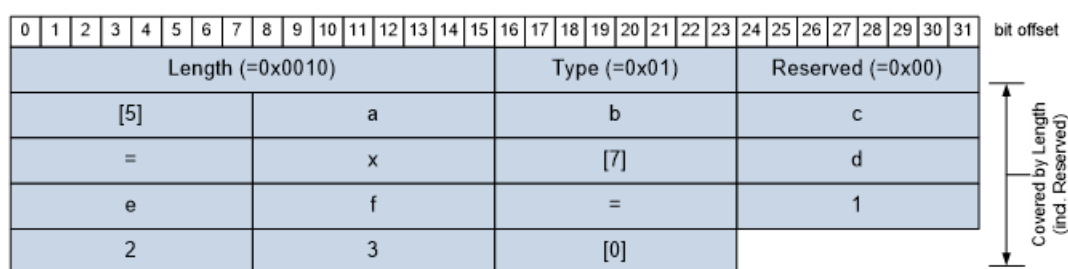


**[SWS\_Sd\_00292]** [The Configuration String shall be constructed as follows from the SdServerCapabilityRecord and SdClientCapabilityRecord (Eventgroups of Services with ID 0xFFFE shall include the Services CapabilityRecord):

- For every SdServerCapabilityRecordKey/ SdServerCapabilityRecordValue or SdClientServiceCapabilityRecordKey/ SdClientServiceCapabilityRecordValue pair:
  - A config\_item\_string is constructed of the concatenation of key, "=", and value.
  - The length of this config\_item\_string is written as uint8 to the configuration string.
  - The config\_item\_string is appended to the configuration string.
- Append a 0x00 uint8 at the end. This means no further config\_item\_string follows.

]()

Example for Configuration Option:



**Figure 7.6: - Example for Configuration Option**

**[SWS\_Sd\_00461]** [SdServerCapabilityRecordValue and SdClientServiceCapabilityRecordValue are allowed to be empty.

This means that after "=" the next length uint8 or "0" follows. ]()

**[SWS\_Sd\_00466]** [Receiving a config\_item\_string without an "=" sign shall be interpreted as key present without value. ]()

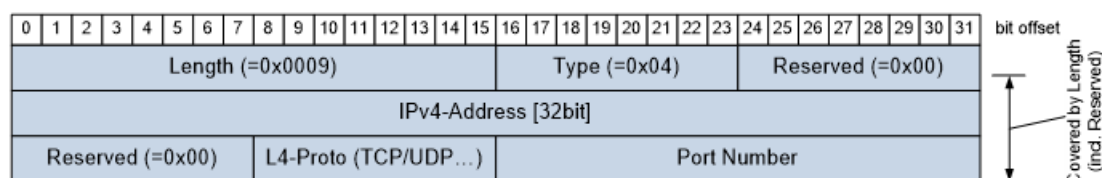
**[SWS\_Sd\_00467]** [Multiple config\_item\_string with the same key in a single configuration option shall be supported. ]()

**[SWS\_Sd\_00468]** [If SdInstanceHostname exists, a key "hostname" with the value set to the string of this configuration item shall be added to the Configuration Option. ]()

### 7.3.9.2 IPv4 Endpoint Option

This chapter describes the fields and values of the IPv4 Endpoint Option, which transports unicast IP Address, Layer 4 Protocols (e.g. UDP or TCP), and Port Number; thus, the information needed to communicate with a service.

When receiving a Service Discovery message offering a service and transporting an IPv4 Endpoint Option, ECUs receiving this message can dynamically configure the Socket Adaptor for using this service by updating a Socket Connection.



**Figure 7.7: - IPv4 Endpoint Option format**

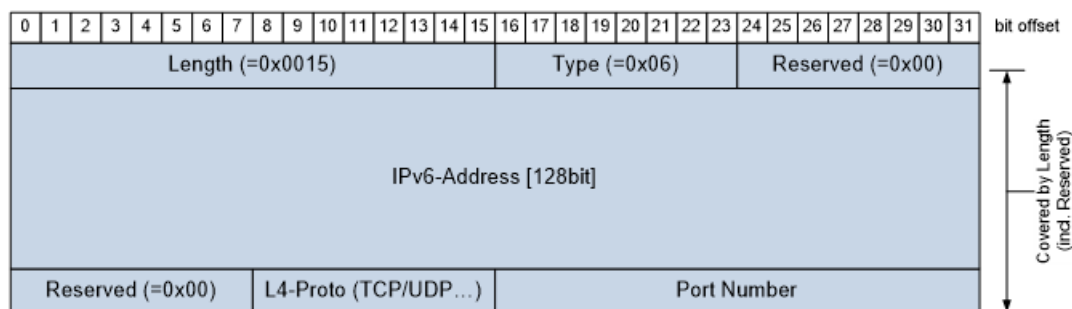
**[SWS\_Sd\_00755]** [The ports shall be used for the events and notification events as well.

- When using UDP the server uses the announced port as source port.
- With TCP the client shall check the status of the socket connection by calling SoAd\_GetSoConMode(). Calling this API has to provide SOAD\_SOCON\_ONLINE state for at the dedicated socket connection.

In addition, if a secure port was selected, an security association needs to be established before sending the subscription. Otherwise events and notification events can neither be sent secure ports nor received.]()

### 7.3.9.3 IPv6 Endpoint Option

This chapter describes the fields and values of the IPv6 Endpoint Option, which is the same as the IPv4 Endpoint Option except that it transport IPv6 Addresses instead IPv4 Addresses.



**Figure 7.8: - IPv6 Endpoint Option format**

**[SWS\_Sd\_00756]** [The ports shall be used for the events and notification events as well.

- When using UDP the server uses the announced port as source port.
- With TCP the client shall check the status of the socket connection by calling SoAd\_GetSoConMode(). Calling this API has to provide SOAD\_SOCON\_ONLINE state for at the dedicated socket connection.

In addition, if a secure port was selected, an security association needs to be established before sending the subscription. Otherwise events and notification events can neither be sent secure ports nor received.

]()

#### 7.3.9.4 IPv4 Multicast Option

The IPv4 Multicast option is either used by an SdServerService to announce its configured Eventhandler multicast endpoint or by a SdClientService to announce its configured Consumed Eventgroup multicast endpoint:

- If it is used as Eventhandler multicast endpoint, then an SdServerService announces the IPv4 multicast address, the transport layer protocol (ISO/OSI layer 4) and the port number, to where the multicast-events and multicast-notification-events are sent to.
- If it is used as Consumed Eventgroup multicast endpoint, then an SdClientService indicates the IPv4 multicast address, the transport layer protocol (ISO/OSI layer 4) and the port number, where the SdClient expects events to be received.

As transport layer protocol, only UDP is supported.

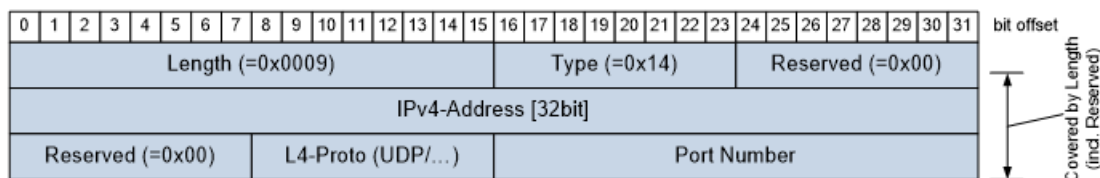


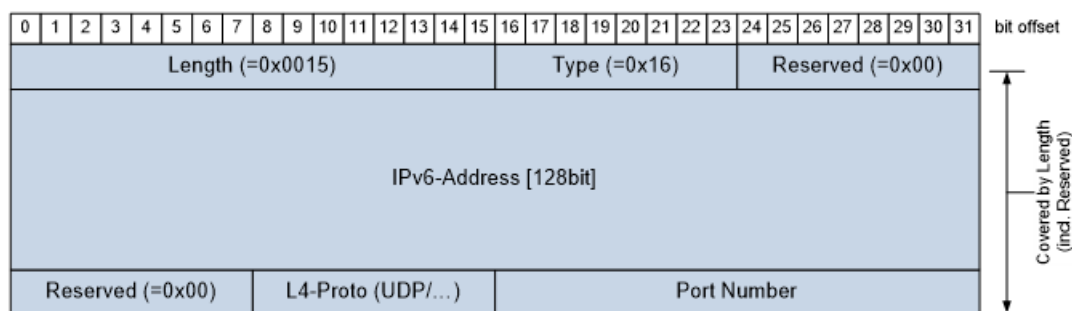
Figure 7.9: - IPv4 Multicast Option format

#### 7.3.9.5 IPv6 Multicast Option

The IPv6 Multicast option is either used by an SdServerService to announce its configured Eventhandler multicast endpoint or by an SdClientService to announce its configured Consumed Eventgroup multicast endpoint:

- If it is used as Eventhandler multicast endpoint, then an SdServerService announces the IPv6 multicast address, the transport layer protocol (ISO/OSI layer 4) and the port number, to where the multicast-events and multicast-notification-events are sent to.
- If it is used as Consumed Eventgroup multicast endpoint, then an SdClientService indicates the IPv6 multicast address, the transport layer protocol (ISO/OSI layer 4) and the port number, where the SdClient expects events to be received.

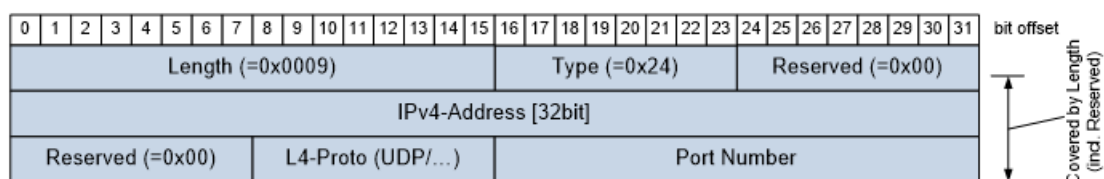
As transport layer protocol, only UDP is supported.



**Figure 7.10: - IPv6 Multicast Option format**

### 7.3.9.6 IPv4 SD Endpoint Option

The IPv4 SD Endpoint Option transports the endpoint (i.e. IP-Address and Port) of the senders SD implementation. This is used to identify the SOME/IP-SD Instance in cases in which the IP-Address and/or Port Number cannot be used.



**Figure 7.11: - IPv4 SD Endpoint Option**

**[SWS\_Sd\_00799] IPv4 Service Discovery Endpoint Options** [The SOME/IP-SD implementation shall support

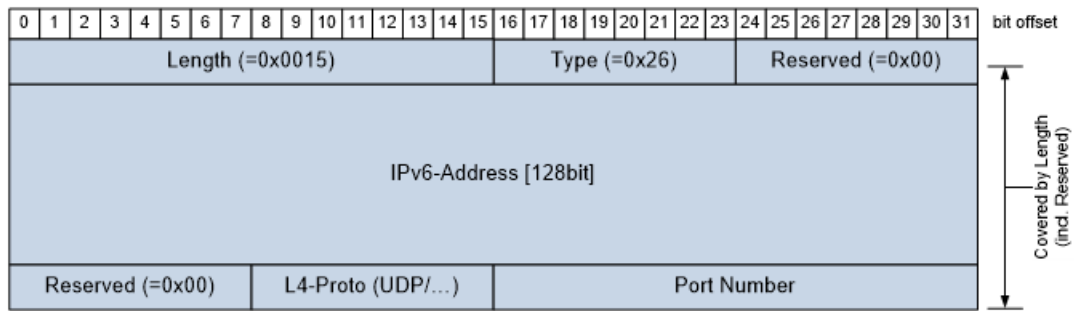
[PRS\_SOMEIPSD\_00547], [PRS\_SOMEIPSD\_00650], [PRS\_SOMEIPSD\_00651],  
[PRS\_SOMEIPSD\_00548], [PRS\_SOMEIPSD\_00549], [PRS\_SOMEIPSD\_00550],  
[PRS\_SOMEIPSD\_00551], [PRS\_SOMEIPSD\_00552], [PRS\_SOMEIPSD\_00856],  
[PRS\_SOMEIPSD\_00857], [PRS\_SOMEIPSD\_00854].

]([RS\\_SOMEIPSD\\_00025](#))

Note: The sending of the SD Endpoint Options is currently out of scope of AUTOSAR.

### 7.3.9.7 IPv6 SD Endpoint Option

The IPv6 SD Endpoint Option transports the endpoint (i.e. IP-Address and Port) of the senders SD implementation. This is used to identify the SOME/IP-SD Instance in cases in which the IP-Address and/or Port Number cannot be used.



**Figure 7.12: - IPv6 SD Endpoint Option**

**[SWS\_Sd\_00800] IPv6 Service Discovery Endpoint Options** [The SOME/IP-SD implementation shall support

[PRS\_SOMEIPSD\_00554], [PRS\_SOMEIPSD\_00654], [PRS\_SOMEIPSD\_00555],  
[PRS\_SOMEIPSD\_00556], [PRS\_SOMEIPSD\_00557], [PRS\_SOMEIPSD\_00558],  
[PRS\_SOMEIPSD\_00559], [PRS\_SOMEIPSD\_00837], [PRS\_SOMEIPSD\_00859],  
[PRS\_SOMEIPSD\_00860], [PRS\_SOMEIPSD\_00855].

]([RS\\_SOMEIPSD\\_00025](#))

Note: The sending of the SD Endpoint Options is currently out of scope of AUTOSAR.

### 7.3.9.8 Handling missing, redundant, and conflicting Options

This section describes the error handling of received options.

**Note:** Several entry types are used in combination with different option types:

- Offer and StopOffer entries use an IPv4 or IPv6 Endpoint Option. The Endpoint Option content (IP address, port and L4-protocol) are identified via SdServerServiceTcpRef and SdServerServiceUdpRef
- Subscribe and StopSubscribe entries use an IPv4 or IPv6 Endpoint Option, if the corresponding Client Service refer to SdClientServiceTcpRef or SdClientServiceUdpRef
- Subscribe and StopSubscribe entries use an IPv4 or IPv6 Multicast Option, if the corresponding Client Service refer to SdClientServiceMulticastRef
- SubscribeEventGroupAck entries use an IPv4 or IPv6 Multicast Option. The Endpoint Option content (multicast IP address and port) are identified via SdMulticastEventSoConRef

For further details on Handling missing, redundant, and conflicting Options, see [6, PRS SOME/IP Service Discovery Protocol] 4.1.4.6 Error Handling

**[SWS\_Sd\_00663]** [SubscribeEventgroup entries shall be answered with SubscribeEventgroupNack, if the SubscribeEventgroup entry reference two or more options that are in conflict or the option type is unknown.]()

**Note:**

For Service Endpoints Options see SdClientServiceTcpRef and SdClientServiceUdp Ref. For Eventgroup Endpoint Options see SdEventActivationRef at SdEventHandlerUdp/SdEventHandlerTcp/SdEventHandlerMulticast.

See also [PRS\_SOMEIPSD\_00231] and [PRS\_SOMEIPSD\_00361].

### 7.3.9.9 Security considerations for Options

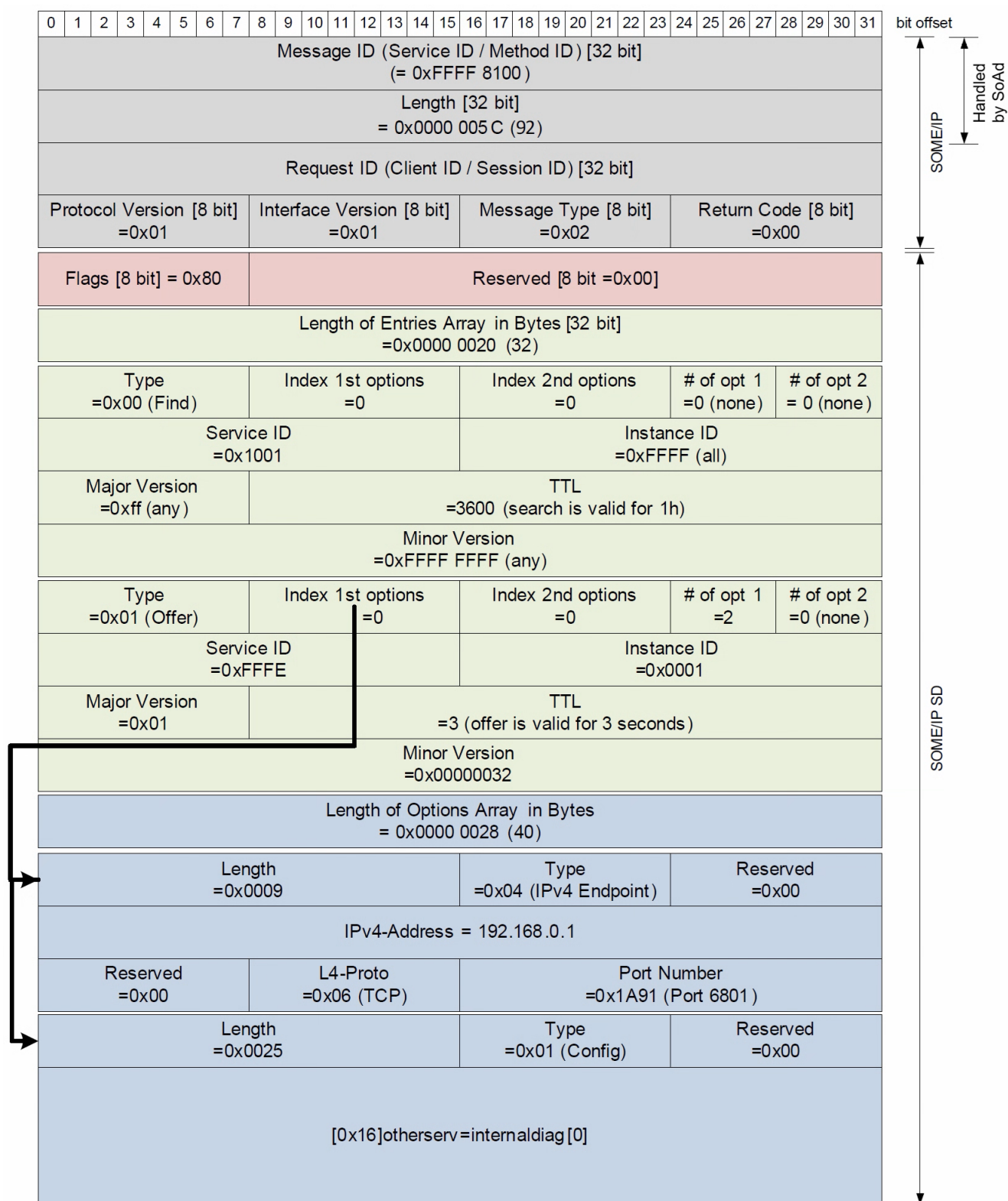
[SWS\_Sd\_00720] [For checking if endpoints are topological correct, the value of [ECUC\_SD\_00128] shall be used in order to determine on how many leading bits shall be compared to check if an IP address is qualified as local. If not present, the value of the locally configured netmask for the IP address shall be used.] ()

### 7.3.10 Entries referencing Options

This chapter describes how Entries can reference two runs of Options with zero to fifteen options each in order to reference additional information.

**Note:** Entries support two option runs to allow referencing the same Options by different Entries. With a single option run, sharing Endpoint Options while having different Configuration Options per Entry would not have work efficiently.

**Note:** [Figure 7.13](#) shows an SD message example, which has an entry referencing two options in the first run:



**Figure 7.13: - Example with Entries referencing Options**

The following table shows which Option is allowed to be carried by different Entries (all other combinations shall not be used):



## 7.4 Service Discovery Entry Types

ECUs shall distribute available Service Instances and Service Instances needed as well as the Eventgroups of these Service Instances. For this purpose, they exchange entries using Service Discovery messages. This chapter describes how these entries are encoded to offer and find services as well as find and subscribe Eventgroups.

### 7.4.1 Entries for Services (common requirements)

These requirements are valid for all Entries concerning Services including Entries of Type 0x00, 0x01, 0x02, and 0x03.

**Note:** Currently only Service Entries of type 0x00 and 0x01 are defined in this specification.

**[SWS\_Sd\_00295]** [An Instance ID of 0xFFFF shall mean any possible instances and are not allowed for OfferService and StopOfferService entries.] ()

**[SWS\_Sd\_00296]** [FindService entries shall carry Service ID, Service Instance ID, Major Version, and Minor Version as configured in SdClientServiceID, SdClientServiceInstanceID, SdClientServiceMajorVersion, and SdClientServiceMinorVersion.] ()

**[SWS\_Sd\_00297]** [OfferService and StopOfferService shall carry Service ID, Service Instance ID, Major Version, Minor Version, and as configured in SdServerServiceID, SdServerServiceInstanceID, SdServerServiceMajorVersion, and SdServerServiceMinorVersion.] ()

**[SWS\_Sd\_00298]** [FindService entries shall carry the TTL as configured in SdClientTimerTTL.]

**Note:** The TTL value for FindService shall be ignored by the server service, and the configuration is only kept for backward compatibility.] ()

**[SWS\_Sd\_00299]** [OfferService entries shall carry the TTL as configured in SdServerTimerTTL.] ()

**[SWS\_Sd\_00267]** [All entries concerning Services (FindService, OfferService and StopOfferService shall carry - i.e. reference - the options as configured.] ()

**Note:** see also chapter [7.3.9.6](#).

### 7.4.2 FindService entry

FindService entries allow finding Service Instances.

For further details on FindService entry, see [\[6, PRS SOME/IP Service Discovery Protocol\]](#) Chapter 4.1.2.5 Service Entries



**[SWS\_Sd\_00503]** [If SdVersionDrivenFindBehavior is set to EXACT\_OR\_ANY\_MINOR\_VERSION, the Service Discovery shall use exact minor version for the FindService entry, which means services with this specific minor version shall only be returned.]()

**[SWS\_Sd\_00752]** [If SdVersionDrivenFindBehavior is set to EXACT\_OR\_ANY\_MINOR\_VERSION the Service Discovery shall use 0xFFFF FFFF (ANY) for the FindService entry, which means that services with any minor version shall be returned]()

**Note to [SWS\_Sd\_00503] and [SWS\_Sd\_00752]:** It is expected that the Minor Version on client side is configured to 0xFFFF FFFF in normal operation since the client should accept all different Minor Versions. Different Minor Versions shall be compatible to each other.

**[SWS\_Sd\_10503]** [If SdVersionDrivenFindBehavior is set to MINIMUM\_MINOR\_VERSION the following points shall be considered by the Service Discovery module:

- the Minor Version shall be set to the minimum acceptable required minor version in the configuration
- Service Discovery shall use 0xFFFF FFFF (ANY) for the FindService entry, which means that services with any minor version shall will be returned

]()

**Note:** This described behavior of [SWS\_Sd\_10503] is differend from [PRS\_SOMEIPSD\_00825]

**Note:** Handling of received services entries, where the SdVersionDrivenFindBehavior is set to MINIMUM\_MINOR\_VERSION is specified in requirement [SWS\_Sd\_04089] of chapter 7.5.3 Receiving Entries

**[SWS\_Sd\_00504]** [TTL shall be set according to the configuration.]()

### 7.4.3 OfferService entry

To offer Service Instances, the OfferService entry shall be used.

For further details on OfferService entry, see [6, PRS SOME/IP Service Discovery Protocol] Chapter 4.1.2.5 Service Entries

**[SWS\_Sd\_00612]** [If the Load Balancing Option is used, the Weight field shall be set to the configured value of SdServerServiceLoadBalancingWeight.]()

**[SWS\_Sd\_00611]** [If the Load Balancing Option is used, the Priority field shall be set to the configured value of SdServerServiceLoadBalancingPriority.]()

#### 7.4.4 Building OfferService entries

**[SWS\_Sd\_00478]** [This chapter describes how to derive all necessary data to assemble an OfferService Message:

1. Derive all static data from the configuration container. These are e.g:
  - Container SdServerService: SdServerServiceId
  - Container SdServerService: SdServerServiceInstanceId
  - Container SdServerService: SdServerServiceMajorVersion
  - Container SdServerService: SdServerServiceMinorVersion
  - Container SdServerTimer: SdServerTimerTTL
  - Container SdInstance: SdInstanceHostname
2. If TCP is configured for this service (configuration item SdServerServiceTcpRef exists):
  - The generator derives a SoConID out of the SoConGroup referenced by the configuration parameter SdServerServiceTcpRef
  - Call the Socket Adaptor's API SoAd\_GetLocalAddr() with the derived SoConID to get back the IP Address, Transport protocol (Layer 4), and the port number needed for the Endpoint Option.
  - Build the relevant Endpoint Option with L4-Protocol set to TCP (shall be same as in LocalAddr) .
3. If UDP is configured for this service (configuration item SdServerServiceUdpRef exists):
  - The generator derives a SoConID out of the SoConGroup referenced by the configuration parameter SdServerServiceUdpRef
  - Call the Socket Adaptor's API SoAd\_GetLocalAddr() with the derived SoConID to get back the IP Address, Transport protocol (Layer 4), and the port number needed for the Endpoint Option.
  - Build the relevant Endpoint Option with L4-Protocol set to TCP (shall be same as in LocalAddr) .
4. Build Configuration Option if configured (see configuration item SdServerCapabilityRecord and SdInstanceHostname).
5. Build OfferService Entry as described above.

]()

### 7.4.5 StopOfferService entry

To stop offering Service Instances, the StopOfferService entry shall be used.

For further details on StopOfferService entry, see [6, PRS SOME/IP Service Discovery Protocol] Chapter 4.1.2.5 Service Entries.

### 7.4.6 Eventgroup Entries (Common requirements)

The following requirements are valid for all Entries concerning Eventgroups including Entries of Type 0x04, 0x05, 0x06, and 0x07.

**Note:** Currently only Eventgroup Entry of Type 0x06 and 0x07 are defined in this specification.

For further details on Eventgroup Entries, see [6, PRS SOME/IP Service Discovery Protocol] Chapter 4.1.3.1 Eventgroup Entry

**[SWS\_Sd\_00289]** [Eventgroups entries include:

- SubscribeEventgroup and StopSubscribeEventgroup
- SubscribeEventgroupAck and SubscribeEventgroupNack

]()

**[SWS\_Sd\_00291]** [Eventgroup entries shall set the Eventgroup ID to the ID of the Eventgroup (configuration parameters SdConsumedEventGroupId and SdEventHandlerEventGroupId).]()

**Note:** Eventgroup ID 0x0000 is reserved.

**[SWS\_Sd\_00301]** [SubscribeEventgroup, and StopSubscribeEventgroup entries shall set the Service IDs, Service Instance IDs, and Eventgroup IDs based on the configuration (configuration parameters SdClientServiceId and SdClientServiceInstanceId).]()

**[SWS\_Sd\_00304]** [SubscribeEventgroup entries shall have the TTL field set to the configured value (configuration parameter SdClientTimerTTL of SdConsumedEventGroup) and the SubscribeEventgroupAck entry shall use the TTL value of the SubscribeEventgroup entry it acknowledges.]()

**[SWS\_Sd\_00307]** [Eventgroup entries shall carry the options as configured.]()

### 7.4.7 SubscribeEventgroup entry

To subscribe to Eventgroups, the SubscribeEventgroup entry shall be used.

For further details on SubscribeEventgroup Entries, see [6, PRS SOME/IP Service Discovery Protocol] Chapter 4.1.3.1 Eventgroup Entry

**[SWS\_Sd\_00693]** [The Counter field in the Type 2 Entry format is used to differentiate different Subscribe Eventgroups to otherwise identical Eventgroups (i.e. same Service ID, same Instance ID, same Eventgroup ID, and same Major Version). The Counter field shall be reflected by the Server to the Subscribe Eventgroup Ack and Nack entries.

If identical Consumed Eventgroups are configured with different Endpoints, then the SD shall use the Counter to differentiate the different Subscriptions. The value of the Counter can be determined by the implementation.]()

**Note:**

A width of 4 bits limits this to 16 different Subscriptions to the same Eventgroup.

**[SWS\_Sd\_00757]** [In case network security protocols are in use clients shall be holding back their SubscribeEventgroup, as long as the security association that enables secure communication is not established (see [\[SWS\\_Sd\\_00761\]](#)).

]()

#### 7.4.8 StopSubscribeEventgroup entry

To stop subscribing to an Eventgroup, the StopSubscribeEventgroup entry shall be used.

For further details on StopSubscribeEventgroup Entries, see [\[6, PRS SOME/IP Service Discovery Protocol\]](#) Chapter 4.1.3.1 Eventgroup Entry

#### 7.4.9 SubscribeEventgroupAck entry

To acknowledge a SubscribeEventgroup entry, the SubscribeEventgroupAck entry shall be used and shall be used with the values as in the SubscribeEventgroup entry it stops.

For further details on SubscribeEventgroupAck Entries, see [\[6, PRS SOME/IP Service Discovery Protocol\]](#) Chapter 4.1.3.1 Eventgroup Entry

#### 7.4.10 SubscribeEventgroupNack entry

For further details on SubscribeEventgroupNack Entries, see [\[6, PRS SOME/IP Service Discovery Protocol\]](#) Chapter 4.1.3.1 Eventgroup Entry

**[SWS\_Sd\_00698]** [If a SubscribeEventgroup entry referencing two conflicting Endpoint Options (UDP or TCP) is received then a SubscribeEventgroupNack shall be generated. Endpoint options are considered conflicting if they are of the same type but hold different values, like different IP or Port number.]()

**[SWS\_Sd\_00758]** [When the client receives a `SubscribeEventgroupNack` as response to a `SubscribeEventgroup` for which a security association is required, the client shall check the state of the security protocol (see [\[SWS\\_Sd\\_00761\]](#)) and shall restart the security protocol, if not yet started (see [\[SWS\\_Sd\\_00465\]](#)).] ()

#### 7.4.11 Building `SubscribeEventgroup` entries

**[SWS\_Sd\_00701]** [This requirement describes how to derive all necessary data to assemble a `SubscribeEventgroup` Message:

1. Derive all static data from the configuration container. These are e.g:
  - Container `SdClientService`: `SdClientServiceId`
  - Container `SdClientService`: `SdClientServiceInstanceId`
  - Container `SdClientService`: `SdClientServiceMajorVersion`
  - Container `SdClientService`: `SdClientServiceMinorVersion`
  - Container `SdConsumedEventGroupTimerRef` - `SdClientTimer`: `SdClientTimerTTL`
  - Container `SdInstance`: `SdInstanceHostname`
2. If TCP is configured for this service (configuration item `SdClientServiceTcpRef` exists):
  - Find the relevant `SocketConnection` based on the `SdClientServiceTcpRef` (finding `SoConGroup`) and the `Endpoint Option` of the `OfferService` entry (finding `SoCon` within).
  - Call the `Socket Adaptor's API SoAd_GetLocalAddr()` with the derived `SoConID` to get back the IP Address, Transport protocol (Layer 4), and the port number needed for the `Endpoint Option`.
  - Build the relevant `Endpoint Option` with L4-Protocol set to TCP (shall be same as in `LocalAddr`).
3. If UDP is configured for this service and used as Consumed Eventgroup unicast endpoint (configuration item `SdClientServiceUdpRef` exists):
  - Find the relevant `SocketConnection` based on the `SdClientServiceUdpRef` (finding `SoConGroup`) and the `Endpoint Option` of the `OfferService` entry (finding `SoCon` within).
  - Call the `Socket Adaptor's API SoAd_GetLocalAddr()` with the derived `SoConID` to get back the unicast IP Address, Transport protocol (Layer 4), and the port number needed for the `Endpoint Option`.
  - Build the relevant `Endpoint Option` with L4-Protocol set to UDP (shall be same as in `LocalAddr`).

4. If UDP is configured for this service and used as Consumed Eventgroup multicast endpoint (configuration item SdClientServiceMulticastRef exists):
  - Find the relevant SocketConnection based on the SdClientServiceMulticastRef (finding SoConGroup) and the Endpoint Option of the OfferService entry (finding SoCon within).
  - Call the Socket Adaptor's API SoAd\_GetLocalAddr() with the derived SoConID to get back the multicast IP Address, Transport protocol (Layer 4), and the port number needed for the Multicast Option.
  - Build the relevant Multicast Option with L4-Protocol set to UDP (shall be same as in LocalAddr).
5. Build Configuration Option if configured (see configuration item SdClientCapabilityRecord and SdInstanceHostname).
6. Build SubscribeEventgroup Entry as described above.

]()

## 7.5 Sending and Receiving of Messages

This chapter describes how messages are transmitted and received using the Socket Adaptor module.

**[SWS\_Sd\_00039]** [The Service Discovery module sends Service Discovery messages (Offer, StopOffer, Find,.. ) using the SoAd\_IfTransmit() API carrying the referenced TxPdu (see configuration parameter SdInstanceTxPdu).]()

**[SWS\_Sd\_00040]** [The Service Discovery module receives Service Discovery messages via the API Sd\_SoAdIfRxIndication() and the configuration items SdInstanceUnicastRxPdu and SdInstanceMulticastRxPdu. The received remote address must be saved in the call context of the Sd\_RxIndication.]()

**[SWS\_Sd\_00479]** [When receiving Service Discovery messages the values of all reserved fields shall be ignored.]()

**[SWS\_Sd\_00708]** [Every time the Service Discovery module receives a SOME/IP-SD message, the consistency of this message has to be checked. This includes but is not limited to:

- Validating that the SOME/IP-SD message is long enough to fit the entries and options arrays (total length = 12 + length of entries array + length of options array).
- Check that entries reference existing options.

In case a malformed message has been received, the extended production error SD\_E\_MALFORMED\_MSG shall be reported.]()

### 7.5.1 Sequence for message transmission

**[SWS\_Sd\_00480]** [This chapter describes the interaction with the Socket Adaptor module to send Service Discovery messages:

1. Precondition: Service Discovery message is assembled
2. In case the message shall be sent via unicast:
  - Call the Socket Adaptor's API `SoAd_SetRemoteAddr`
3. In case the message shall be sent via multicast:
  - Call the API `SoAd_SetRemoteAddr` to set the destination
4. Call `SoAd_IfTransmit()` to send the message on the bus

Please also refer to the sequence "CLIENT/SERVER: TransmitSdMessage" shown in Chapter 9.]()

**Note:**

This can be achieved for example by checking the status of all Service Instances and Eventgroups cyclically and afterwards assembling the Service Discovery Messages.

**[SWS\_Sd\_00651]** [The amount of separate Service Discovery messages shall be reduced, i.e.: Combine as much information as possible into one Service Discovery message before calling the Socket Adaptor's transmit API. This means that when a entry is sent after waiting the appropriate delay (i.e. based on Request-Response-Delay) all other entries for this communication partner may be packed into the Service Discovery message as well.]()

### 7.5.2 Sequence for message reception

**[SWS\_Sd\_00482]** [This chapter describes the interaction with the Socket Adaptor on how Service Discovery messages are received:

1. When the SocketAdaptor receives a Service Discovery message, the API `Sd_RxIndication()` is called.
2. Using the indicated `RxPduld`, the associated `SoConId` for this SD Instance has to be determined.
3. Call API `SoAd_GetRemoteAddr()` with this `SoConId`.
4. Store address and message for further processing.
5. Reset the `SoCon` back to Wildcard using `SoAd_ReleaseRemoteAddr()`
6. The entries shall be processed exactly in the order they arrived.

Please also refer to the sequence "CLIENT/SERVER: Sd\_RxIndication" shown in Chapter 9.]()



**Note:**

For deriving the SoConId, the SoAdSocketRoute corresponding to this RxPdId should refer either to a SoAdSocketConnection or to a SoAdSocketConnectionGroup containing a single SoAdSocketConnection.

**[SWS\_Sd\_00696]** [If the entries of a single Service Discovery Message would lead to closing and opening the same Socket Connection in the Socket Adaptor, the Service Discovery shall not close the Socket Connection first.] ()

**Note:** Closing and opening Socket Connections (especially with TCP), conflicts with the behavior of the Service Discovery and leads to suboptimal reaction times.

### 7.5.3 Receiving Entries

When receiving entries the relevant Service Instance or Eventgroups have to be identified, which is explained in this section.

**[SWS\_Sd\_00488]** [If SdClientServiceMinorVersion is set to 0xFFFFFFFF and SdVersionDrivenFindBehavior is set to EXACT\_OR\_ANY\_MINOR\_VERSION, the Minor Version in a received OfferService or StopOfferService entry is not checked for identifying Service Instances and its associated Eventgroups.] ()

**[SWS\_Sd\_00489]** [If SdClientServiceMinorVersion is set to any value except 0xFFFFFFFF and SdVersionDrivenFindBehavior is set to EXACT\_OR\_ANY\_MINOR\_VERSION, the Minor Version in a received OfferService or StopOfferService shall be checked for identifying Service Instances and its associated Eventgroups. The Service Discovery module shall process a OfferService or StopOfferService where the minor version of the received entry match exact the configured minor version of the corresponding SdClientService.] ()

**Note:**

We call each configured service instance fulfilling the SWS items [\[SWS\\_Sd\\_00488\]](#) and [\[SWS\\_Sd\\_00489\]](#) a service instance match candidate.

**[SWS\_Sd\_04089]** [If SdVersionDrivenFindBehavior is set to MINIMUM\_MINOR\_VERSION, the Minor

Version in a received OfferService or StopOfferService shall be checked for identifying Service Instances and its associated Eventgroups. The Service Discovery module shall process a OfferServices or StopOfferServices where the minor version of the received entry are equal or greater than the configured minor version of the corresponding SdClientService.] ()

**Note:** This described behavior of [\[SWS\\_Sd\\_04089\]](#) is differend from [\[PRS\\_SOMEIPSD\\_00825\]](#)



**[SWS\_Sd\_07016]** [If a service match candidate is detected for a ClientService where SdVersionDrivenFindBehavior is set to MINIMUM\_MINOR\_VERSION and the ClientService has already triggered a subscription to another ServerService, the Service Discovery module shall silently discard this service match candidate.]()

**[SWS\_Sd\_01503]** [The Service Discovery module shall ignore all received service entries of a Client Service, where the minor version of the received entry is specified within a version blocklist of the corresponding SdClientService (see SdBlocklistedVersions).]()

**[SWS\_Sd\_00716]** [If either the received Type 1 SD entry references a configuration option or a service match candidate has capability records configured (i.e., SdServerCapabilityRecord in case of a received FindService entry or SdClientCapabilityRecord in case of a OfferService or a StopOfferService entry), the configured SdCapabilityRecordMatchCallout shall be invoked by the SD implementation.]()

**[SWS\_Sd\_00717]** [A received Type 2 SD entry with Service ID 0xFFFE (Non-SOMEIP) shall be matched accordingly to [\[SWS\\_Sd\\_00716\]](#) with the capability records of the Service (SdServerCapabilityRecord in case of a received SubscribeEventgroup or StopSubscribeEventgroup entry or SdClientCapabilityRecord in case of SubscribeEventgroupAck or SubscribeEventgroupNack entry).]()

**[SWS\_Sd\_00718]** [If the invoked SdCapabilityRecordMatchCallout returns true, the respective service instance match candidate actually provides a match for the received SD message including the configured capability records.]()

**[SWS\_Sd\_00719]** [If the invoked SdCapabilityRecordMatchCallout returns false, the respective service instance match candidate actually does not provide a match for the received SD message due to the mismatch with respect to the configured capability records.]()

### **7.5.3.1 Answering behaviour, if receiving Service Discovery Entries via Multicast address**

When receiving Service Discovery messages using multicast, these messages may be received by multiple ECUs at once and multiple ECUs may answer to such a message in parallel. This could lead to overload situations of the ECU which sent the Service Discovery message via multicast, if all receiving ECUs answer in a similar point in time. In order to avoid a high workload on ECU which sent the Service Discovery message via multicast, the answers of the receiving ECUs could delay answer as described in this section.

**[SWS\_Sd\_00491]** [Answers to Entries received via multicast shall be delayed based on the appropriate configuration items:

- For ServerServices:

- SdServerTimerRequestResponseMinDelay
- SdServerTimerRequestResponseMaxDelay
- For ConsumedEventgroups:
  - SdClientTimerRequestResponseMinDelay
  - SdClientTimerRequestResponseMaxDelay

]()

**[SWS\_Sd\_00492]** [The configuration parameters for delaying OfferService entries as response to FindService entries received by multicast shall be taken from the Timer containers referenced by the Service container:

- SdServerService

]()

**[SWS\_Sd\_00493]** [The configuration parameters for delaying SubscribeEventgroup entries as response to OfferService entries received by multicast shall be taken from the Timer containers referenced by the Eventgroup containers:

- SdConsumedEventGroup

]()

**[SWS\_Sd\_00494]** [There shall be a random delay between the appropriate MinDelay and MaxDelay before answering to an Entry received via multicast.]()

**[SWS\_Sd\_00724]** [If SdServerTimerRequestResponseMinDelay and SdServerTimerRequestResponseMaxDelay are set to the same value, this value shall be used as delay.

If SdServerTimerRequestResponseMinDelay and SdServerTimerRequestResponseMaxDelay are set to 0, no delay shall be introduced.]()

**[SWS\_Sd\_00725]** [If SdClientTimerRequestResponseMinDelay and SdClientTimerRequestResponseMaxDelay are set to the same value, this value shall be used as delay.

If SdClientTimerRequestResponseMinDelay and SdClientTimerRequestResponseMaxDelay are set to 0, no delay shall be introduced.]()

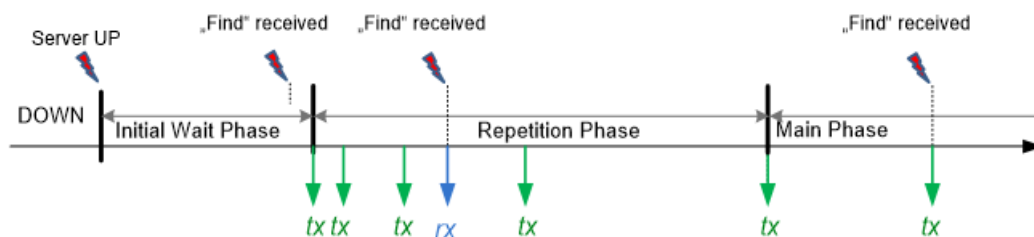
**[SWS\_Sd\_00495]** [Delayed answering Entries received via multicast (as in [\[SWS\\_Sd\\_00494\]](#)) shall no influence other timers (e.g. for handling the Repetition Phase).]()

## 7.6 Timings and repetitions for Server Service and Event Handlers

Especially after starting multiple ECUs, the multicast messages of the Service Discovery come with the risk of overflowing ECUs with too many messages. Therefore, the Service Discovery can be configured with a suitable message sending behavior.

For every Server Service Instance different phases are defined as shown in [Figure 7.14](#):

- Down
- Available
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase



**Figure 7.14: - Communication phases Server**

**[SWS\_Sd\_00605]** [When the Down Phase is entered (coming from states other than init), the API SoAd\_CloseSoCon() shall be called for all Socket Connections associated with this Server Service Instance.]()

**[SWS\_Sd\_00760]** [If a Sd server receives a SubscribeEventgroup entry and client end point is assigned to a socket connection, the server shall call SoAd\_IsConnection-Ready() for this socket connection and client endpoint:

- If the function returns TCPIP\_E\_OK, the server shall respond with SubscribeEventgroupAck.
- For all other return values the server shall discard the entry and respond with SubscribeEventgroupNack.

]()

### 7.6.1 Initial Wait Phase for Server Services

This chapter describes the behavior of the Service Discovery in regard of a Server Service Instance in the Initial Wait Phase.

**[SWS\_Sd\_00317]** [If the following conditions apply, the Initial Wait Phase for this configured Server Service Instance shall be entered:

- Sd\_Init() has been called
- SdServerService state was set to SD\_SERVER\_SERVICE\_AVAILABLE (via Sd\_ServerServiceSetState() or Sd\_ServiceGroupStart())
- Sd\_LocalIpAddrAssignmentChg() with state "TCPIP\_IPADDR\_STATE\_ASSIGNED" has been called for the first IpAddrId associated with the SdInstance TxPdu.

]()

**Note:** Service Discovery expects that the IP address of the data/control path to be always the same. This means that a call of Sd\_LocalIpAddrAssignmentChg() affects the control path and data path simultaneously.

**[SWS\_Sd\_00330]** [When the Initial Wait Phase is entered, the routing of the Server Service shall be enabled. See SdServerServiceActivationRef of this Server Service Instance.]()

**[SWS\_Sd\_00318]** [When entering the Initial Wait Phase, a random timer shall be started, using a random value within the configured range of SdServerTimerInitialOfferDelayMin and SdServerTimerInitialOfferDelayMax.]()

**[SWS\_Sd\_00320]** [If a SubscribeEventgroup Entry or StopSubscribeEventgroup Entry are received within the Initial Wait Phase (or other phases) for an Event Handler of this Server Service Instance, it shall only be processed within the Service Discovery.]()

**Note to [SWS\_Sd\_00320]:** Please refer to the according sequence diagrams and section 7.6.4.

**[SWS\_Sd\_00321]** [

When the calculated random timer based on the min and max values SdServerTimerInitialOfferDelayMin and SdServerTimerInitialOfferDelayMax expires and SoAd\_GetSoConMode() provides SOAD\_SOCON\_ONLINE or SOAD\_SOCON\_RECONNECT state for at least one of the associated socket connection of this service ( configured in SdServerServiceTcpRef or SdServerServiceUdpRef ) :

- OfferService Entry shall be sent.
- If the SdServerTimerInitialOfferRepetitionsMax >0, enter the Repetition Phase
- If the SdServerTimerInitialOfferRepetitionsMax =0, enter the Main Phase.

]()

**Note:**

1. Init Wait Phase could be extended depends upon the out parameter of type SoAd\_SoConModeType provided by SoAd\_GetSoConMode() API.

2. In some case SoAd may need more time to change the socket connection state from SOAD\_SOCON\_OFFLINE to SOAD\_SOCON\_RECONNECT or SOAD\_SOCON\_ONLINE. E.G. Socket Connection will not change to SOAD\_SOCON\_RECONNECT or SOAD\_SOCON\_ONLINE only if InitWaitPhase of Service is configured as 0 or SoAd main function period is greater then Sd main function period.

**[SWS\_Sd\_00323]** [If SdServerService is set to a state other than SD\_SERVER\_SERVICE\_AVAILABLE ( via Sd\_ServerServiceSetState() or Sd\_ServiceGroupStop() ) while being in Initial Wait Phase:

- Enter the Down Phase.
- Set all associated EventHandler to SD\_EVENT\_HANDLER\_RELEASED and report it to the BswM by calling the API BswM\_Sd\_EventHandlerCurrentState.
- Cancel all relevant timers for service instance (see [\[SWS\\_Sd\\_00318\]](#)).

]()

**[SWS\_Sd\_00325]** [If Sd\_LocalIpAddrAssignmentChg() is called with a state other than "TCPIP\_IPADDR\_STATE\_ASSIGNED" while being in Initial Wait Phase, this phase shall be left and the Down Phase shall be entered.]()

**[SWS\_Sd\_00606]** [When the Initial Wait Phase is entered, the API SoAd\_OpenSocket() shall be called for all Socket Connections associated with this Server Service Instance.]()

**Note:** As soon as an IP address is assigned again and no SD\_SERVER\_SERVICE\_DOWN was received, the Initial Wait Phase shall be reentered with the random timer reset to the random value.

## 7.6.2 Repetition Phase for Server Services

This chapter describes the timing behavior of the Service Discovery in regard of Server Service Instances in the Repetition Phase.

**[SWS\_Sd\_00329]** [If the Repetition Phase is entered, the Service Discovery shall wait SdServerTimerInitialOfferRepetitionBaseDelay and send an OfferService Entry.]()

**[SWS\_Sd\_00336]** [After the amount of cyclically sent OfferServices within the Repetition Phase equals the amount of SdServerTimerInitialOfferRepetitionsMax, the Main Phase shall be entered.]()

### Note:

Additionally sent OfferService messages which have been triggered by received Find Service messages shall have no influence on the counter value of the cyclically Offer Service messages.

**[SWS\_Sd\_00331]** [In the Repetition Phase up to SdServerTimerInitialOfferRepetitionsMax OfferService Entries shall be sent with doubling intervals (BaseDelay, first Offer

Service Entries, 2x BaseDelay, second OfferService Entries, 4x BaseDelay, third Offer Service Entries).]()

**Note:** Example config and resulting behavior:

SdServerTimerInitialOfferRepetitionBaseDelay=30

SdServerTimerInitialOfferRepetitionsMax=3

*[Initial Wait Phase starts]*

Wait Initial Wait Delay based on Configured Min and Max

Send entry.

*[Initial Wait Phase ends]*

*[Repetition Phase starts]*

Wait 30ms (=30ms \* 20).

Send entry.

Wait 60ms (=30ms \* 21).

Send entry.

Wait 120ms (=30ms \* 22).

Send entry.

*[Repetition Phase ends]*

**Note:** Currently this specification does not allow sending "FindService Entries" using unicast. For compatibility reasons receiving such entries shall be supported.

**[SWS\_Sd\_00333]** [If the Service Discovery Module receives a "SubscribeEventgroup" entry, the following step(s) shall be performed in the following order:

- Send a SubscribeEventgroupAck / Nack entry using Unicast considering the appropriate delay without changing the current counter value and without influencing the current running repetition timer.
- Call the BswM with the API BswM\_Sd\_EventHandlerCurrentState() with state SD\_EVENT\_HANDLER\_REQUESTED only if the state for this EventHandler changed (i.e. has not been SD\_EVENT\_HANDLER\_REQUESTED)
- Start the TTL timer according to the value received via the SubscribeEventgroup Entry.

]()

**Note to:** [\[SWS\\_Sd\\_00333\]](#):

- Currently this specification does not allow sending "SubscribeEventgroup Entries" using multicast. For compatibility reasons receiving such entries shall be supported.
- For more details on sending a SubscribeEventgroupAck / Nack entry using Unicast considering the appropriate delay, see Chapter 7.5.3)

**[SWS\_Sd\_00334]** [If the Service Discovery Module receives a StopSubscribeEventgroup Entry, the following step(s) shall be performed in the following order:

- Stop the TTL timer for this client
- Update State
- If this has been the last subscribed client, report "SD\_EVENT\_HANDLER\_RELEASED" to the BswM by calling the API BswM\_Sd\_EventHandlerCurrentState().

]()

**[SWS\_Sd\_00458]** [If the TTL of a received SubscribeEventgroup Entry expires, the following step shall be performed in the following order:

- If this has been the last subscribed client, report "SD\_EVENT\_HANDLER\_RELEASED" to the BswM by calling the API BswM\_Sd\_EventHandlerCurrentState() and update the state within the Service Discovery Module

]()

**[SWS\_Sd\_00338]** [If a ServerService is set to a state other than SD\_SERVER\_SERVICE\_AVAILABLE (i.e. SD\_SERVER\_SERVICE\_DOWN ) (via Sd\_ServerServiceSet State() or Sd\_ServiceGroupStop() ) while being in Repetition Phase:

- Leave this phase and enter the Down Phase.
- Sent a StopOfferService.
- All associated EventHandler which state is not SD\_EVENT\_HANDLER\_RELEASED shall be changed to SD\_EVENT\_HANDLER\_RELEASED and indicated to the BswM by calling the API BswM\_Sd\_EventHandlerCurrentState().

]()

**[SWS\_Sd\_00340]** [If Sd\_LocalIpAddrAssignmentChg() is called with a state other than "TCPIP\_IPADDR\_STATE\_ASSIGNED" while being in Repetition Phase, this phase shall be left and the Down Phase shall be entered.]()

**[SWS\_Sd\_00732]** [If the Service Discovery Module is in Repetition Phase and a TCP-based socket connection has been lost (i.e. Socket connection is other than SOAD\_SOCON\_ONLINE), the Service Discovery Module shall call:

- SoAd\_ReleaseRemoteAddr()
- SoAd\_DisableSpecificRouting()
- SoAd\_CloseSoCon()



- delete all internally stored subscriptions belonging to that connection.
- SoAd\_OpenSoCon()

The Service Discovery Module shall stay in the Repetition Phase. ]()

**[SWS\_Sd\_00341]** [When the state SD\_SERVER\_SERVICE\_DOWN is set by Sd\_ServerServiceSetState() or Sd\_ServiceGroupStop() in Repetition Phase, the routing of this Server Service Instance shall be disabled. See SdServerServiceActivationRef of this Server Service Instance. ]()

### 7.6.3 Main Phase for Server Services

**[SWS\_Sd\_00342]** [The Service Discovery Module shall stay in the Main Phase for the configured Server Service as long as the following conditions apply:

- Server Service is in state "SD\_SERVER\_SERVICE\_AVAILABLE" ( indicated by a call of Sd\_ServerServiceSetState() or Sd\_ServiceGroupStart())
- IP address is assigned and can be used (i.e. Sd\_LocalIpAddrAssignmentChg has been called with status TCPIP\_IPADDR\_STATE\_ASSIGNED)

]()

**[SWS\_Sd\_00449]** [If SdServerTimerOfferCyclicDelay is greater than 0, in the Main Phase an OfferService entry shall be sent cyclically with an interval defined by configuration item SdServerTimerOfferCyclicDelay. ]()

**[SWS\_Sd\_00450]** [The first OfferService is sent SdServerTimerOfferCyclicDelay after the beginning of the Main Phase. ]()

**[SWS\_Sd\_00451]** [If SdServerTimerOfferCyclicDelay is 0, no OfferService entries shall be sent in Main Phase for this Server Service Instance. ]()

**[SWS\_Sd\_00343]** [If the Service Discovery Module receives a FindService Entry the following step shall be performed:

- Send an "OfferService Entry" considering the appropriate delay.

]()

**Note:** Currently this specification does not allow sending "FindService Entries" using unicast. For compatibility reasons receiving such entries shall be supported.

**Note to [SWS\_Sd\_00343]:** For more details on sending an "OfferService Entry" considering the appropriate delay, see Chapter 7.5.3)

**[SWS\_Sd\_00344]** [If the Service Discovery Module receives a "SubscribeEventgroup", the following step(s) shall be performed in the following order:

- Send a SubscribeEventgroupAck / Nack entry using Unicast considering the appropriate delay without influencing the current running main phase timer.



- Report to the BswM SD\_EVENT\_HANDLER\_REQUESTED by calling the API BswM\_Sd\_EventHandlerCurrentState().
- Start the TTL timer according to the value received via the "SubscribeEventgroup".

]()

**Note:** Currently this specification does not allow sending "SubscribeEventgroup Entries" using multicast. For compatibility reasons receiving such entries shall be supported.

**Note to [SWS\_Sd\_00344]:** For more details on sending a SubscribeEventgroupAck / Nack entry using Unicast considering the appropriate delay, see Chapter 7.5.3)

**[SWS\_Sd\_00345]** [If the Service Discovery Module receives a "StopSubscribeEventgroup", the following step(s) shall be performed in the following order:

- Stop the TTL timer and remove it from the notification list
- If no other client is subscribed to this Eventgroup anymore, enter the State "SD\_EVENT\_HANDLER\_RELEASED" and report it to the BswM by calling the API BswM\_Sd\_EventHandlerCurrentState () with state "SD\_EVENT\_HANDLER\_RELEASED".

]()

**[SWS\_Sd\_00347]** [If the API LocalIpAddrAssignmentChg has been called with a state other than TCPIP\_IPADDR\_STATE\_ASSIGNED,

- The Service Discovery Module shall leave the Main Phase and enter the DOWN Phase
- All EventHandler which are not in state SD\_EVENT\_HANDLER\_RELEASED shall be set to SD\_EVENT\_HANDLER\_RELEASED and be indicated to the BswM module by calling the API BswM\_Sd\_EventHandlerCurrentState

]()

**[SWS\_Sd\_00733]** [If a TCP-based socket connection has been lost (i.e. Socket connection is other than SOAD\_SOCON\_ONLINE), the Service Discovery Module shall call:

- SoAd\_ReleaseRemoteAddr()
- SoAd\_DisableSpecificRouting()
- SoAd\_CloseSoCon()
- delete all internally stored subscriptions belonging to that connection.
- SoAd\_OpenSoCon()

The Service Discovery Module shall stay in the Main Phase.]()

**[SWS\_Sd\_00348]** [If a SdServerService is set to state "SD\_SERVER\_SERVICE\_DOWN" (indicated by a call of Sd\_ServerServiceSetState() or Sd\_ServiceGroupStop()) while the IP address is still assigned (i.e. Sd\_LocalIpAddrAssignmentChg has been called with state TCPIP\_IPADDR\_STATE\_ASSIGNED), the Service Discovery module shall

- send a StopOfferService
- enter the DOWN Phase
- all subscriptions of the eventgroup(s) of this service instance shall be deleted and SD\_EVENT\_HANDLER\_RELEASED and reported to BswM using the API BswM\_Sd\_EventHandlerCurrentState

]()

**[SWS\_Sd\_00349]** [When the Main Phase is left, the routing of this Server Service Instance shall be disabled. See SdServerServiceActivationRef of this Server Service Instance.]()

**[SWS\_Sd\_00403]** [When the TTL timer (contained in TTL field find or Subscribe entry) expires in state "SD\_EVENT\_HANDLER\_REQUESTED",

enter the state SD\_EVENT\_HANDLER\_RELEASED and report it to the BswM by calling the BswM\_Sd\_EventHandlerCurrentState().]()

#### 7.6.4 Fan out control

This chapter describes the interaction between Service Discovery and Socket Adaptor (SoAd) in order to configure the TX path for sending out events (fan out). It has to be considered, that a SdClientService could either subscribe with an Consumed Eventgroup unicast endpoint (transferred within a Endpoint Option) or with a Consumed Eventgroup multicast endpoint (transferred within a Multicast Option).

**[SWS\_Sd\_00452]** [The Service Discovery shall keep track of the subscribed clients per Event Handler and remove clients from the fan out, if the last SubscribeEventgroup entry was longer ago than the time specified in its TTL field of that SubscribeEventgroup entry. This shall be handled independently if the client subscribed with a Consumed Eventgroup unicast endpoint, Consumed Eventgroup multicast endpoint or if the Event Handler has set SdEventHandlerMulticastThreshold to 1 (Events are transmitted exclusively via Eventhandler multicast endpoint)]()

**Note:** Service Discovery has to maintain the TTL time per subscribed Client Service Instance independent if the client subscribed with a Consumed Eventgroup unicast endpoint or Consumed Eventgroup multicast endpoint or if the affected SdServerService transmit its Events via the Evenhandler multicast endpoint according to the configuration of SdEventHandlerMulticastThreshold. In any case the Server Service Instance

must know its subscribed clients with respect to the unicast remote address (IP and port) of the client.

**[SWS\_Sd\_00453]** [If SdEventHandlerTCP is configured: For every SubscribeEventgroup entry of this Event Handler and the SubscribeEventgroup entry reference an Endpoint Option, the following shall be done:

- The relevant Routing Groups shall be identified by SdEventHandlerTcp.
- The relevant TCP Socket Connection of this client shall be identified using the Address/Port of Endpoint Option (TCP) referenced in the SubscribeEventgroup entry and the SdServerServiceTcpRef, or shall be set up, if not existed before.
- Check state of incoming TCP connection using SoAd\_GetSoConMode:
  - If the return mode is SOAD\_SOCON\_ONLINE, then perform the following actions:
    - \* If the client was not subscribed before receiving the aforementioned entry
      - call SoAd\_EnableSpecificRouting with SdEventActivationRef and the Socket Connection.
      - call SoAd\_IfSpecificRoutingGroupTransmit with SdEventTriggering Ref and the Socket Connection.
    - \* Answer this SubscribeEventgroup by using SubscribeEventgroupAck entry.
  - If the returned mode is other than SOAD\_SOCON\_ONLINE, then this shall be handled according to [\[SWS\\_Sd\\_00732\]](#) or [\[SWS\\_Sd\\_00733\]](#) (depending on current service state) and answer this SubscribeEventgroup by using SubscribeEventgroupNack.

]()

**[SWS\_Sd\_00454]** [

If SdEventHandlerUdp is configured: For every SubscribeEventgroup entry of this Eventhandler and if the SubscribeEventgroup entry references a Unicast Endpoint Option, the following shall be done:

- The relevant Routing Groups shall be identified by SdEventHandlerUdp.
- If the relevant UDP Socket Connection of this client shall be identified using the Eventgroup unicast endpoint (Address/Port) of Endpoint Option (UDP) referenced in the SubscribeEventgroup entry and the SdServerServiceUdpRef, or shall be set up (SoAd\_SetUniqueRemoteAddr()), if not existed before.
  - If no Wildcard Socket Connection is left, SD\_E\_OUT\_OF\_RES shall be reported.
- Only if the client was not subscribed before receiving this entry:

- SoAd\_EnableSpecificRouting with SdEventActivationRef and the Socket Connection depending on current number of subscribed clients with different endpoint information and the SdEventHandlerMulticastThreshold.
- SoAd\_IfSpecificRoutingGroupTransmit with SdEventTriggeringRef and the Socket Connection.

]()

**[SWS\_Sd\_00753]** [If SdEventHandlerUdp is configured: For every SubscribeEventgroup entry of this Eventhandler and if the SubscribeEventgroup entry references a Multicast Option, the following shall be done:

- The relevant Routing Groups shall be identified by SdEventHandlerUdp.
- The relevant UDP Socket Connection of this client shall be identified using the Eventgroup multicast endpoint (Address/Port) of the Multicast Option referenced in the SubscribeEventgroup entry and the SdServerServiceUdpRef, or shall be set up (SoAd\_SetUniqueRemoteAddr()), if not existed before.
  - If no Wildcard Socket Connection is left, SD\_E\_OUT\_OF\_RES shall be reported.
- The following action shall be performed, if no other client has already subscribed with the same Consumed Eventgroup Multicast endpoint information:
  - Call SoAd\_EnableSpecificRouting with SdEventActivationRef and the corresponding Socket Connection. The corresponding Socket Connection shall be the configured Socket Connection referenced by SdMulticastEventSoConRef, if the number of subscribed clients with different endpoint information has reached SdEventHandlerMulticastThreshold. Otherwise the identified Socket Connection (described in the previous point)
- Only if the client was not subscribed before receiving this entry:
  - SoAd\_IfSpecificRoutingGroupTransmit with SdEventTriggeringRef and the Socket Connection.

]()

**Note:**

- SdClientServices which subscribe with the same Consumed Eventgroup multicast endpoint, share the same SoAdSocketConnection on SdServerService side
- A SdServiceService could send the same event at the same time to a Consumed Eventgroup unicast endpoint or Consumed Eventgroup multicast endpoint. This is announced within the SubscriptionEventgroup entry which could reference either a IPv4/IPv6 Endpoint option (unicast endpoint) or via IPv4/IPv6 Multicast option (multicast endpoint).
- Transmission of initial Events (SdEventTriggeringRef is configured) in combination with a subscription using a Consumed Eventgroup multicast endpoint has

to be used carefully. This has to be ensured by the network communication design. Rational: every subscription to the same Consumed Eventgroup multicast endpoint would trigger a transmission of an initial Event, which is received by all currently subscribed Clients. This could cause misbehavior for communication for example which use sequence counters (e.g. E2E communication).

**[SWS\_Sd\_00754]** [Each Eventhandler shall qualify based on the configured SdEventHandlerMulticastThreshold and the number of clients with different endpoint information (either received as Eventgroup unicast endpoint or as Eventgroup multicast endpoint), if the threshold has been reached to transmit the Events via the configured Eventhandler multicast endpoint (see SdMulticastEventSoConRef).]()

**[SWS\_Sd\_00455]** [The number of subscribed clients with different endpoint information shall be used to control when to enable/disable Consumed Eventgroup unicast or Consumed Eventgroup multicast connection, or when to enable/disable Eventhandler Multicast connection by calling SoAd\_EnableSpecificRouting and SoAd\_DisableSpecificRouting:

- If SdEventHandlerMulticastThreshold = 0: Setup a Consumed Eventgroup unicast connection or a Consumed Eventgroup multicast connection to every subscribed client (please note: Eventhandler Multicast connection is always disabled).
- If SdEventHandlerMulticastThreshold = 1: Setup a Eventhandler Multicast connection if one or more clients are subscribed (please note: Consumed Eventgroup unicast connections and Consumed Eventgroup multicast connections are always disabled).
- If SdEventHandlerMulticastThreshold > 1:
  - Setup a Consumed Eventgroup unicast connection or a Consumed Eventgroup multicast connection for all subscribed clients if the number of subscribed clients with different endpoint information < SdEventHandlerMulticastThreshold,
  - else setup a Eventhandler Multicast connection and switch automatically based on the number of subscribed clients with different endpoint information:
    - \* If the number of subscribed clients with different endpoint information is larger or equal than the threshold, then the Eventhandler multicast connection shall be used for transmission.
    - \* If the number of subscribed clients with different endpoint information is smaller than the threshold, the individual Consumed Eventgroup unicast connections and Consumed Eventgroup multicast connections shall be used for transmission.

]()

Example:

- Precondition
  - Server\_Service\_A contain Eventgroup\_A with SdEventHandlerMulticastThreshold = 3
  - Server\_Service\_A.Eventgroup\_A has Multicast endpoint configured to EMc\_endpoint\_A
  - All Clients subscribe to ServerService\_A.Eventgroup\_A
- Example 1:
  - Client\_A subscribe with unicast endpoint
  - Client\_B and Client\_C subscribe with the same multicast endpoint
  - Result:
    - \* SdEventHandlerMulticastThreshold has NOT reached
    - \* Correspondings Events of ServerService\_A.Eventgroup\_A transmitted to Client\_A via unicast endpoint and to Client\_B and Client\_C via the same multicast endpoint
- Example 2:
  - Client A subscribe with unicast endpoint A
  - Client B subscribe with unicast endpoint B
  - Client C subscribe with multicast endpoint C
  - Result:
    - \* SdEventHandlerMulticastThreshold has reached
    - \* Corresponding Events of ServerService A.Eventgroup\_A are transmitted via Multicast endpoint EMc\_endpoint\_A to Client\_A, Client\_B, Client\_C

### 7.6.5 Sharing of SdServerTimer

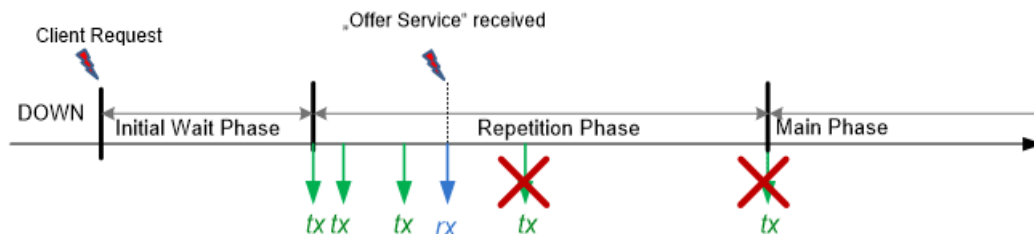
**[SWS\_Sd\_00743]** [If several ServerServices refer to the same SdServerTimer, they shall share a common timer (and therefore a common random offset), if they either refer to the same SdServiceGroup and do not refer to any other (additional) SdService Group or, if SdServerServiceAutoAvailable of all ServerServices are set to TRUE.]()

## 7.7 Timings and repetitions for Client Service and Consumed Eventgroups

The Service Discovery phases allow minimizing the number of Service Discovery messages sent while allowing for very fast synchronization upon ECU start.

This de-emphasis is realized by the following Phases:

- Down
- Requested
  - Initial Wait Phase
  - Repetition Phase
  - Main Phase



**Figure 7.15: - Communication phases Client**

**[SWS\_Sd\_00761]** [If a Sd client wants to subscribe to an Eventgroup, it shall call SoAd\_IsConnectionReady() for the assigned socket connection and indented server end point.

- If the function returns TCPIP\_E\_OK, the client can send a SubscribeEventgroup entry.
- If the function returns TCPIP\_E\_PENDING, the client shall delay the subscription.
- For all other return values the client shall discard the subscription request.

]()

### 7.7.1 Down Phase for Client Services

**[SWS\_Sd\_00462]** [As long as a service is not requested by the BswM, the Service Discovery shall not send FindService Entry entries.]()

**[SWS\_Sd\_00463]** [If an OfferService Entry is received during Down Phase,

- The Service Discovery shall store the state of this Service instance.
- A timer shall be set/reset to the TTL value of the received OfferService entry (TTL timer).



- Until the TTL Timer expires or a StopOfferService entry is received, the Service instance is considered Available.

]()

**[SWS\_Sd\_00464]** [If a SdClientService is set to state SD\_CLIENT\_SERVICE\_REQUESTED (by call of Sd\_ClientServiceSetState() or Sd\_ServiceGroupStart()) while being in Down Phase:

- If no OfferService entry was received before or its TTL timer expired already:
  - The Initial Wait Phase shall be entered,
- If an OfferService entry was received and its TTL timer did not expire yet:
  - The Main Phase shall be entered.
  - \* Further processing shall be done according to [\[SWS\\_Sd\\_00721\]](#).

]()

## 7.7.2 Initial Wait Phase for Client Services

This chapter describes the behavior of the Service Discovery in regard of a Client Service Instance in the Initial Wait Phase.

**[SWS\_Sd\_00350]** [If the following conditions apply, the Initial Wait Phase for this configured Client Service Instance shall be entered:

- Sd\_Init() has been called.
- SdClientService was set to state SD\_CLIENT\_SERVICE\_REQUESTED (indicated by a call of Sd\_ClientServiceSetState() or Sd\_ServiceGroupStart() or Sd\_ClientServiceAutoRequired = TRUE)
- Sd\_LocalIpAddrAssignmentChg() with state "TCPIP\_IPADDR\_STATE\_ASSIGNED" has been called for the first IpAddrId associated with the SdInstance TxPdu.

]()

**[SWS\_Sd\_00351]** [This Client Service Instance shall stay in the Initial Wait Phase for a time within the configured range of SdClientTimerInitialFindDelayMin and SdClientTimerInitialFindDelayMax unless an OfferService entry for this Client Service Instance is received or this random timer expires.]()

**[SWS\_Sd\_00352]** [If an OfferService Entry for this Client Service Instance is received within the Initial Wait Phase,

- The calculated random timer, which has been started when entering the Initial Wait Phase, shall be canceled.
- Leave the Initial Wait Phase and enter the Main Phase.



- Further processing shall be done according to [SWS\_Sd\_00721].

]()

**[SWS\_Sd\_00353]** [When the calculated random timer based on the parameters SdClientTimerInitialFindDelayMin and SdClientTimerInitialFindDelayMax expires (i.e. no OfferService has been received within this timespan), the following shall be done in the following order:

- FindService Entry shall be sent.
- If the SdClientTimerInitialFindRepetitionsMax>0, enter the Repetition Phase
- If the SdClientTimerInitialFindRepetitionsMax=0, enter the Main Phase

]()

**[SWS\_Sd\_00355]** [If a SdClientService is set to state SD\_CLIENT\_SERVICE\_RELEASED (by call of Sd\_ClientServiceSetState() or Sd\_ServiceGroupStop()) while being in Initial Wait Phase, this phase shall be left and the Service shall enter Down Phase.]()

**[SWS\_Sd\_00456]** [If for any reasons the Initial Wait Phase is left, the calculated random timer (of the Initial Wait Phase) for this Service Instance shall be stopped.]()

**[SWS\_Sd\_00357]** [If Sd\_LocalIpAddrAssignmentChg() is called with a state other than "TCPIP\_IPADDR\_STATE\_ASSIGNED" while being in Initial Wait Phase, the Down Phase shall be entered.]()

**[SWS\_Sd\_00354]** [If the API Sd\_Init() is called while being in Initial Wait Phase, the Down Phase shall be entered.]()

### 7.7.3 Repetition Phase for Client Services

**[SWS\_Sd\_00358]** [When the Repetition Phase is entered, the Service Discovery Module shall start the timer SdClientTimerInitialFindRepetitionsBaseDelay.]()

**[SWS\_Sd\_00457]** [When the timer SdClientTimerInitialFindRepetitionsBaseDelay expires within the Repetition Phase, a FindOffer Message shall be sent.]()

**[SWS\_Sd\_00363]** [In the Repetition Phase up to SdClientTimerInitialFindRepetitionsMax FindServer entries shall be sent with doubling intervals (BaseDelay, first FindService Entry, 2x BaseDelay, second FindService Entry, 4x BaseDelay, third FindService Entry,...).]()

**Note:** Example config and resulting behavior (no OfferService received during example):

SdClientTimerInitialFindRepetitionBaseDelay=30

SdClientTimerInitialFindRepetitionMax=3

[Initial Wait Phase starts]

Wait Initial Wait Delay based on Configured Min and Max

Send entry.

[Initial Wait Phase ends]

[Repetition Phase starts]

Wait 30ms (=30ms \* 20).

Send entry.

Wait 60ms (=30ms \* 21).

Send entry.

Wait 120ms (=30ms \* 22).

Send entry.

[Repetition Phase ends]

**[SWS\_Sd\_00365]** [If the Service Discovery Module receives an OfferService Entry in the Repetition Phase while the current state SD\_CLIENT\_SERVICE\_REQUESTED is for this Client Service Instance, the following step(s) shall be performed in the following order:

- Cancel the repetition timer.
- Leave the Repetition Phase immediately and enter the Main Phase.
  - Further processing shall be done according to [\[SWS\\_Sd\\_00721\]](#).

]()

**[SWS\_Sd\_00751]** [If the Service Discovery Module receives an StopOfferService Entry while the current state SD\_CLIENT\_SERVICE\_REQUESTED is for this Client Service Instance, the following step(s) shall be performed in the following order:

- Cancel the repetition timer.
- Leave the Repetition Phase immediately and enter the Main Phase.

]()

**[SWS\_Sd\_00369]** [After sending the maximum amount of repetitions (defined by SdClientTimerInitialFindRepetitionsMax) of FindService entries, the Repetition Phase shall be left and the Main Phase shall be entered.]()

**[SWS\_Sd\_00371]** [If SdClientService is set to state SD\_CLIENT\_SERVICE\_RELEASED (by call of Sd\_ClientServiceSetState() or Sd\_ServiceGroupStop()) while being in Repetition Phase, this phase shall be left and the service instance shall enter Down Phase.]()

**[SWS\_Sd\_00373]** [If `Sd_LocallpAddrAssignmentChg()` is called with a state other than `"TCPIP_IPADDR_STATE_ASSIGNED"` while being in Repetition Phase the Down Phase shall be entered.]()

**[SWS\_Sd\_00730]** [If the TCP/IP connection has been lost (Socket connection is other than `SOAD_SOCON_ONLINE`), the Service Discovery Module shall leave the Repetition Phase, enter the Initial Wait Phase, and stop the TTL timers of the associated Client Service Instances and EventGroups.]()

#### 7.7.4 Main Phase for Client Services

**[SWS\_Sd\_00375]** [The Service Discovery Module shall stay in the Main Phase as long as the following conditions apply:

- Client Service was set to state `"SD_CLIENT_SERVICE_REQUESTED"` (indicated by a call of `Sd_ClientServiceSetState()` or `Sd_ServiceGroupStart()`)
- IP address assigned and can be used (i.e. `Sd_LocallpAddrAssignmentChg` has been called with status `TCPIP_IPADDR_STATE_ASSIGNED`).

]()

**[SWS\_Sd\_00721]** [If the Service Discovery Module receives an OfferService Entry, the following step(s) shall be performed in the following order:

- If received TTL is not equal to the max value, update the timer by the received TTL value.
- The relevant UDP/TCP Socket Connection of this service shall be identified using the Address and Port of the Endpoint Option referenced in the Offer entry of this service or shall be set up (`SoAd_SetUniqueRemoteAddr()`), if not existed before.
  - If no Socket Connection is left, `SD_E_OUT_OF_RES` shall be reported.
- Open UDP/TCP connection with API `SoAd_OpenSoCon()` if `SdClientServiceUdpRef` / `SdClientServiceTcpRef` is configured and was not opened before.
- Check state of the associated Socket Connections of this Client Service using `SoAd_GetSoConMode()`.
  - If the associated Socket Connections are in state `SOAD_SOCON_ONLINE` in the Main phase:
    - \* If the client service has not been reported as `SD_CLIENT_SERVICE_AVAILABLE`:
      - the API `SoAd_EnableSpecificRouting()` shall be called with `SdClientServiceActivationRef` (see `SdConsumedMethods`) and the relevant Socket Connections for this Client Service Instance.

- SD\_CLIENT\_SERVICE\_AVAILABLE shall be indicated to the BswM module by calling the API `BswM_Sd_ClientServiceCurrentState()`.
- \* For each currently requested Consumed Eventgroup of this Client Service Instance (Consumed Eventgroups are requested using `Sd_ConsumedEventGroupSetState()` and with state SD\_CONSUMED\_EVENTGROUP\_REQUESTED or automatically on startup if `Sd_ConsumedEventGroupAutoRequire` is configured to true), the following shall be done in exactly this order:
  - StopSubscribeEventgroup entry shall be sent out, if the last SubscribeEventgroup entry was sent as reaction to an OfferService entry received via Multicast, it was never answered with a SubscribeEventgroupAck, and the current OfferService entry was received via Multicast.
  - A SubscribeEventgroup entry shall be sent out.  
 If `SdSubscribeEventgroupRetryEnable` is set to TRUE and if `SdSubscribeEventgroupRetryMax` is greater 0, the Eventgroup subscription retry counter shall be reset to 1.

]()

#### Note:

Refer to [SWS\_Sd\_00702], [SWS\_Sd\_00703] and [SWS\_Sd\_00704] for the enabling of routing groups. The transmission of a response to an Offer received via multicast shall be delayed with the configured delay. When the request response delay elapses before the associated Socket Connections are in state SOAD\_SOCON\_ONLINE, the StopSubscribeEventgroup and SubscribeEventgroup shall be delayed until the Socket Connections are online and shall not be considered as reaction to an OfferService entry received via Multicast. When the request response delay elapses while the Client Service is in state RELEASED, there shall be no response to this Offer entry. The amount of separate Service Discovery messages shall be reduced, i.e.: Combine as much information as possible into one Service Discovery message before calling the Socket Adaptor's transmit API.

**[SWS\_Sd\_00722]** [When the Client Service is reported as SD\_CLIENT\_SERVICE\_DOWN to the BswM by calling the API `BswM_Sd_ClientServiceCurrentState()`

- the API `SoAd_DisableSpecificRouting()` shall be called with `SdClientServiceActivationRef` (see `SdConsumedMethods`) and the relevant Socket Connections for this Client Service Instance.

]()

**[SWS\_Sd\_00695]** [If a StopSubscribeEventgroup and SubscribeEventgroup for the same Eventgroup (i.e. same Service ID, Instance ID, Eventgroup ID, Counter, and Major Version) have to be sent out, these entries have to be directly after each other in the same SD message (no entry between them).]()

**[SWS\_Sd\_00377]** [If the Service Discovery Module receives a SubscribeEventgroupAck fitting a Consumed Eventgroup that is not yet available, the following steps shall be performed in the following order:

- If the SubscribeEventgroupAck references a Multicast Endpointoption
  - The relevant Socket Connection Group shall be identified using SdConsumedEventGroupMulticastGroupRef with the local Address and Port of the Multicast Endpoint Option or set one up using SoAd\_RequestIpAddrAssignment().
  - If SdSetRemAddrOfClientRxMulticastSoCon is set to TRUE, the relevant Socket Connection of this service shall be identified using the Address and Port of the Endpoint Option referenced in the Offer entry of this service or shall be set up (SoAd\_SetUniqueRemoteAddr()), if not existed before.
    - \* If no Wildcard Socket Connection is left, SD\_E\_OUT\_OF\_RES shall be reported.
  - If SdSetRemAddrOfClientRxMulticastSoCon is set to FALSE, a Wildcard Socket Connection of this service shall be used without updating the according remote Address, i.e. Wildcard of this Socket Connection shall be kept.
    - \* If no Wildcard Socket Connection is left, SD\_E\_OUT\_OF\_RES shall be reported.
  - The relevant Routing Group shall be identified by following SdConsumedEventGroupMulticastActivationRef.
  - Call SoAd\_EnableSpecificRouting() with the SocketID and the Routing GroupID .
- Call BswM\_Sd\_ConsumedEventGroupCurrentState with SD\_CONSUMED\_EVENTGROUP\_AVAILABLE if the datapath was set up successfully.
- Setup the TTL timer with the TTL of the SubscribeEventgroupAck entry if the datapath was set up successfully.

]()

**[SWS\_Sd\_00465]** [If a Service Discovery Message contains only a SubscribeEventgroupNack entry but no SubscribeEventgroupAck entry for the same Eventgroup, Service Discovery shall do the following:

- Report the DEM error SD\_E\_SUBSCR\_NACK\_RECV (see [\[ECUC\\_SD\\_00123\]](#))
- If SdClientServiceTcpRef is configured for this service, or if SoAd\_IsConnectionReady() returned a different value than TCPIP\_E\_OK, determine the used SoCon and call the API SoAd\_CloseSoCon() with the SoConID and parameter abort set to TRUE

- If `SdClientServiceTcpRef` is configured for this service, or if `SoAd_IsConnection-Ready()` returned a different value than `TCPIP_E_OK`, determine the used `SoCon` and call the API `SoAd_OpenSoCon()` with the `SoConID`.

]()

**[SWS\_Sd\_00367]** [If the Service Discovery Module receives a `StopOfferService` Entry, the following step(s) shall be performed in the following order:

- Stop the TTL timers of this Client Service Instance and all related Consumed Eventgroups.
- Report this Client Service as `DOWN` if it was reported `AVAILABLE` before (call `BswM_Sd_ClientServiceCurrentState` with `SD_CLIENT_SERVICE_DOWN` and the Client Service's handle ID).
- Report all Consumed Eventgroups as `DOWN` that were reported `AVAILABLE` before (call `BswM_Sd_ConsumedEventGroupCurrentState` with `SD_CONSUMED_EVENTGROUP_DOWN` and the Consumed Eventgroup's handle ID).
- If `SdSubscribeEventgroupRetryEnable` is set to `TRUE` and if `SdSubscribeEventgroupRetryMax` is greater 0, cancel the corresponding client service subscription retry delay timer and reset subscription retry counter of all corresponding Eventgroups to 0.
- Close all Socket Connections associated with this Client Service Instance that have been opened before.
- Stay in Main Phase and do not send `FindService` entries.

]()

**[SWS\_Sd\_00741]** [If a Consumed Eventgroup switches to the state `SD_CONSUMED_EVENTGROUP_REQUESTED` while the corresponding state of the requested Service Instance was already set to `SD_CLIENT_SERVICE_AVAILABLE`

(due to an already received Offer Service with TTL `0xFFFFFFFF`), a `SubscribeEventgroup` entry shall be sent out only if all of the following conditions apply:

- `SdSubscribeEventgroupRetryEnable` is set to `TRUE`,
- `SdSubscribeEventgroupRetryMax` is greater 0,

]()

**Note:**

Requirement [\[\[SWS\\_Sd\\_00741\]\]](#) ensures that a Client can still subscribe to Eventgroups at any point in time when it is needed, even though cyclic Offers of the corresponding `ServerService` are not present in the main phase (`SdServerTimerOfferCyclicDelay` set to 0). In this case, no cyclic Offer is needed for triggering the transmissions of `SubscribeEventgroup` entries.

**[SWS\_Sd\_00712]** [If Sd\_LocalIpAddrAssignmentChg() is called with a state other than "TCPIP\_IPADDR\_STATE\_ASSIGNED" while being in Main Phase:

- The Down Phase shall be entered.
- "SD\_CLIENT\_SERVICE\_DOWN" shall be indicated to the BswM module by calling the API BswM\_Sd\_ClientServiceCurrentState(), if the present state is SD\_CLIENT\_SERVICE\_AVAILABLE.
- "SD\_CONSUMED\_EVENTGROUP\_DOWN" shall be indicated to the BswM module by calling the API BswM\_Sd\_ConsumedEventGroupCurrentState() for all associated ConsumedEventgroups, if the present state is SD\_CONSUMED\_EVENTGROUP\_AVAILABLE.
- If SdSubscribeEventgroupRetryEnable is set to TRUE and if SdSubscribeEventgroupRetryMax is greater 0, cancel the corresponding client service subscription retry delay timer and reset subscription retry counter of all corresponding Eventgroups to 0.

]()

**[SWS\_Sd\_00731]** [If the TCP/IP connection has been lost (Socket connection is other than SOAD\_SOCON\_ONLINE), the Service Discovery Module shall leave the Main Phase, enter the Initial Wait Phase, and stop the TTL timers of the associated Client Service Instances and EventGroups.]()

**[SWS\_Sd\_00380]** [The Service Discovery Module shall leave the Main Phase and enter the state SD\_CLIENT\_SERVICE\_DOWN if at least one of the listed conditions described in [SWS\_Sd\_00375] does not apply any more.]()

**[SWS\_Sd\_00381]** [If a SdClientService is set to state "SD\_CLIENT\_SERVICE\_RELEASED" (indicated by a call of Sd\_ClientServiceSetState() or Sd\_ServiceGroupStop()) while all other conditions listed in [SWS\_Sd\_00375] still apply, the Service Discovery module shall perform the following steps:

- Enter the Down Phase and indicate the state SD\_CLIENT\_SERVICE\_DOWN to the BswM by calling the API BswM\_Sd\_ClientServiceCurrentState ().
- For all subscribed eventgroups of this Client Service,
  - a StopSubscribeEventgroup shall be sent
  - the status shall be set to SD\_CONSUMED\_EVENTGROUP\_DOWN and reported to BswM by calling the API BswM\_Sd\_ConsumedEventGroupCurrentState().
- If SdSubscribeEventgroupRetryEnable is set to TRUE and if SdSubscribeEventgroupRetryMax is greater 0, cancel the corresponding client service subscription retry delay timer and reset subscription retry counter of all corresponding Eventgroups to 0.

]()



**[SWS\_Sd\_00713]** [If the Consumed Event Group is not requested anymore as indicated by a call of Sd\_ConsumedEventGroupSetState with state SD\_CONSUMED\_EVENTGROUP\_RELEASED, the Service Discovery module shall perform the following steps for the consumed event group:

- A StopSubscribeEventgroup shall be sent.
- The status shall be set to SD\_CONSUMED\_EVENTGROUP\_DOWN and be reported to the BswM by calling the API BswM\_Sd\_ConsumedEventGroupCurrentState(), if the status is not currently SD\_CONSUMED\_EVENTGROUP\_DOWN.
- If SdSubscribeEventgroupRetryEnable is set to TRUE and if SdSubscribeEventgroupRetryMax is greater 0, cancel the corresponding client service subscription retry delay timer and reset subscription retry counter of all corresponding Eventgroups to 0.

]()

**[SWS\_Sd\_00600]** [If the TTL Timer of a Client Service expires, the Service Discovery module shall perform the following steps:

- Enter the Initial Wait Phase and indicate the state SD\_CLIENT\_SERVICE\_DOWN to the BswM by calling the API BswM\_Sd\_ClientServiceCurrentState ().
- All subscribed Eventgroups of this Client Service shall expired in this instance (stop TTL timer) and the expiration shall be handled as describe in [\[SWS\\_Sd\\_00601\]](#).

]()

**[SWS\_Sd\_00601]** [If the TTL Timer of an Eventgroup expires, the Service Discovery module shall perform the following step(s):

- the status shall be set to SD\_CONSUMED\_EVENTGROUP\_DOWN and reported to BswM by calling the API BswM\_Sd\_ConsumedEventGroupCurrentState().

]()

**[SWS\_Sd\_00382]** [When the Main Phase is left,

- The API SoAd\_DisableSpecificRouting() shall be called for all Socket Connections associated with this Client Service ID that have been opened before.
- Close all Socket Connections associated with this Client Service Instance that have been opened before.

]()



### 7.7.5 Fan in control

This section describes the interaction between Service Discovery and Socket Adaptor (SoAd) to configure the RX path for receiving events (fan in).

**[SWS\_Sd\_00702]** [If SdConsumedEventGroupTcpActivationRef is configured: When sending SubscribeEventgroup entries for this Eventgroup, the following shall be done:

- The relevant Routing Group shall be identified by following SdConsumedEventGroupTcpActivationRef.
- The relevant TCP Socket Connection shall be identified by SdClientServiceTcpRef.
- A TCP Endpoint option shall be constructed with these parameters.
- Only if this client is currently not subscribed yet:
  - SoAd\_EnableSpecificRouting with the two parameters above.

]()

**[SWS\_Sd\_00703]** [If SdConsumedEventGroupUdpActivationRef is configured: When sending SubscribeEventgroup entries for this Eventgroup, the following shall be done:

- The relevant Routing Group shall be identified by following SdConsumedEventGroupUdpActivationRef.
- The relevant TCP Socket Connection shall be identified by SdClientServiceUdpRef.
- A UDP Endpoint option shall be constructed with these parameters.
- Only if this client is currently not subscribed yet:
  - SoAd\_EnableSpecificRouting with the two parameters above.

]()

**[SWS\_Sd\_00704]** [If SdConsumedEventGroupMulticastActivationRef is configured: When receiving SubscribeEventgroupAck entries for this Eventgroup and with a referenced Multicast Option, the following shall be done if this client is currently not subscribed yet:

- The relevant Routing Group shall be identified by following SdConsumedEventGroupMulticastActivationRef.
- The relevant UDP Socket Connection shall be identified:
  - Find the relevant Socket Connection Group using SdConsumedEventGroupMulticastGroupRef with the local Address and Port of the Multicast Option or set one up.
  - Find the relevant Socket Connection in this Socket Connection Group by finding the Address and Port of this Services Endpoint or set one up.

- SoAd\_EnableSpecificRouting with the two parameters above.

]()

**[SWS\_Sd\_00711]** [Routing Groups of EventGroups (see SdConsumedEventGroup TcpActivationRef, SdConsumedEventGroupUdpActivationRef, and SdConsumedEventGroupMulticastActivationRef)

shall be deactivated, if they are not needed anymore (Main phase was left, StopOffer received or ConsumedEventgroup was released).]()

**[SWS\_Sd\_00706]** [Every wildcard socket connection shall be reset to wildcard using ReleaseSoAd\_RemoteAddr() if all of the following conditions apply:

- The remote address of the socket connection has been set by SD according to [\[SWS\\_Sd\\_00377\]](#).
- No Eventgroup Subscription for this socket connection is used anymore.

]()

**[SWS\_Sd\_00734]** [Every wildcard socket connection group shall be reset to wildcard using SoAd\_ReleaseIpAddrAssignment() if all of the following conditions apply:

- Local address of the socket connection group has been set by SD according to [\[SWS\\_Sd\\_00377\]](#).
- All socket connections of this socket connection group have been released.

]()

### 7.7.6 Sharing of SdClientTimer

**[SWS\_Sd\_00744]** [If several ClientServices refer to the same SdClientTimer, they shall share a common timer (and therefore a common random offset), if they either refer to the same SdServiceGroup and do not refer to any other (additional) SdServiceGroup or, if SdClientServiceAutoRequire of all ClientServices are set to TRUE.]()

## 7.8 Handling of SdServiceGroups

### 7.8.1 SdServiceGroup definitions

For a SdServiceGroup the following rules apply:

1. A SdClientService and SdServerService, respectively, can belong to any SdServiceGroup.
2. A SdClientService and SdServerService, respectively, is requested and available, respectively, if it belong to a started SdServiceGroup (see [\[SWS\\_Sd\\_00745\]](#)). If

a SdClientService and SdServerService, respectively, does not belong to any SdServiceGroup, the SdClientService and SdServerService, respectively, has to be requested and set to available via Sd\_ServerServiceSetState() or Sd\_ClientServiceSetState() explicitly (see [SWS\_Sd\_00746]).

3. SdClientServices and SdServerServices of different SdInstances could reference the same SdServiceGroup

**Note:**

Rules 1 and 3 are supported by the ServiceDiscovery configuration.

**[SWS\_Sd\_00745]** [A SdClientService and SdServerService, respectively, is requested and available, respectively, if at least one SdServiceGroup is started it refers to.]()

**Note:**

It is expected that the complete state handling of SdServiceGroup is done outside of the AUTOSAR ServiceDiscovery module, e.g. within the Basic Software Mode Manager. In case of a state change, the module that managing the SdServiceGroup states consistently starts or stops the SdServiceGroup via Sd\_ServiceGroupStart() and Sd\_ServiceGroupStop().

The state of SdClientServiceS and SdServerServiceS that are NOT reference any SdServiceGroup can be changed only via a direct call of Sd\_ClientServiceSetState and Sd\_ServerServiceSetState, respectively.

**[SWS\_Sd\_00746]** [The state of a SdClientService and a SdServerService, respectively, which refer to at least one SdServiceGroup shall only be changed via Sd\_ServiceGroupStart and Sd\_ServiceGroupStop, respectively. The state of a SdClientService and SdServerService, respectively, which do NOT reference any SdServiceGroup, shall only be changed via Sd\_ClientServiceSetState() and Sd\_ServerServiceSetState(), respectively.]()

**[SWS\_Sd\_00747]** [The AUTOSAR ServiceDiscovery module shall keep track of requests and availabilities per SdClientServiceS and SdServerServiceS, respectively, which reference at least one SdServiceGroup. Therefore each affected SdClientService and SdServerService shall have a client request counter and server availability counter, respectively. Each time Sd\_ServiceGroupStart() is called, the client request counter shall be incremented for all affected SdClientServices and the server availability counter shall be incremented for all affected SdServerServices. Each time Sd\_ServiceGroupStop() is called the client request counter shall be decremented for all affected SdClientServices, and the server availability counter shall be decremented for all affected SdServerServices.]()

### 7.8.1.1 Initialization of SdServiceGroupS

**[SWS\_Sd\_00748]** [By default, all SdServiceGroupS shall be in the state stopped and they shall not be started automatically by a call to Sd\_Init.]()

### 7.8.1.2 Starting of SdServiceGroupS

By default all SdServiceGroupS are stopped, see [\[SWS\\_Sd\\_00748\]](#). A call to Sd\_ServiceGroupStart() starts a SdServiceGroup if it was previously stopped.

**[SWS\_Sd\_00749]** [If an SdServiceGroup is started by Sd\_ServiceGroupStart(), the AUTOSAR Service Discovery module shall set all SdClientServiceS which are referencing the affected SdServiceGroup to SD\_CLIENT\_SERVICE\_REQUESTED and all SdServerServiceS which are referencing the affected SdServiceGroup to SD\_SERVER\_SERVICE\_AVAILABLE.]()

### 7.8.1.3 Stopping of SdServiceGroupS

A call to Sd\_ServiceGroupStop() stops an SdServiceGroup, if it was previously started.

**[SWS\_Sd\_00750]** [If an SdServiceGroup is stopped by Sd\_ServiceGroupStop(), the AUTOSAR Service Discovery module shall set all SdClientServiceS, which are referencing the affected SdServiceGroup to SD\_CLIENT\_SERVICE\_RELEASED where the corresponding client request counter (see [\[SWS\\_Sd\\_00747\]](#)) has reached 0, and all SdServerServices which are referencing the affected SdServiceGroup to SD\_SERVER\_SERVICE\_DOWN where the corresponding server availability counter (see [\[SWS\\_Sd\\_00747\]](#)) has reached 0.]()

## 7.9 SOME/IP-ACL

The ACL "Access Control List" introduces the possibility of limiting service access to listed authenticated communication partners. It is an effective way to minimize the damage if one of the communication partners is hacked because this communication partner cannot go beyond the services he could provide or consume in his healthy state.

This can be reached by following steps:

Authentication: This is a pre-condition and not the scope of this chapter, the identity of the communication partner shall be authenticated by e.g. TCAM rules Port-based authentication, IPsec, or MACsec. In the case of IPsec or MACsec, Function SoAd\_IsConnectionReady shall be used to make sure that the configured security association has been established.

Authorization: This is the main focus of this chapter, it is about allowing or blocking providing or consuming services based on an access control list (ACL) containing the IP addresses of the allowed service providers and consumers.

## 7.9.1 ACL Configuration

**[SWS\_Sd\_00763]{DRAFT}** [In case of (`SdEnableAclPolicyCheck` is `TRUE`), SD shall activate the ACL feature.]([SRS\\_Eth\\_00164](#))

**[SWS\_Sd\_00764]{DRAFT}** [In ClientService, the ACL `SdClientServiceAllowedProvider` shall be configured with the allowed service provider's IP address to enable the ACL check.]([SRS\\_Eth\\_00164](#))

**[SWS\_Sd\_00765]{DRAFT}** [In ClientService, if the ACL `SdClientServiceAllowedProvider` is not configured, the ACL check shall be disabled for this ClientService.]([SRS\\_Eth\\_00164](#))

**[SWS\_Sd\_00766]{DRAFT}** [In ServerService, the ACL `SdServerServiceAllowedConsumers` shall be configured with the allowed service consumers' IP addresses to enable the ACL check.]([SRS\\_Eth\\_00164](#))

**[SWS\_Sd\_00767]{DRAFT}** [In ServerService, if the ACL `SdServerServiceAllowedConsumers` is not configured, the ACL check shall be disabled for this ServerService.]([SRS\\_Eth\\_00164](#))

**Note:** The ACL contents will be the IP addresses of the allowed communication partners, so this concept will not be applicable in case of dynamic IP via DHCP is used.

### 7.9.1.1 ACL update

**[SWS\_Sd\_00768]{DRAFT}** [The ACL contents shall be updatable via function `Sd_AclUpdate` call with parameters Service ID `SdServiceId`, ServiceInstanceID, the IP address `RemoteAddrPtr` and the required action `RequestType`.]([SRS\\_Eth\\_00165](#))

**[SWS\_Sd\_00769]{DRAFT}** [If function `Sd_AclUpdate` is called with `RequestType` `SD_ACL_ADD_PROVIDER` and the SD has a ClientService with service ID equals `SdServiceId` and service Instance ID equals `ServiceInstanceID`, SD shall update the IP address in `SdClientServiceAllowedProvider` of this ClientService with the IP address in `RemoteAddrPtr`.]([SRS\\_Eth\\_00165](#))

**[SWS\_Sd\_00780]{DRAFT}** [If function `Sd_AclUpdate` is called with `RequestType` `SD_ACL_ADD_CONSUMER` and the SD has a ServerService with service ID equals `SdServiceId` and service Instance ID equals `ServiceInstanceID`, SD shall add the IP address in `RemoteAddrPtr` to this ServerService ACL `SdServerServiceAllowedConsumers`.]([SRS\\_Eth\\_00165](#))

**[SWS\_Sd\_00781]{DRAFT}** [If function Sd\_AclUpdate is called with RequestType SD\_ACL\_REMOVE\_PROVIDER and the SD has a ClientService with service ID equals SdServiceId and service Instance ID equals ServiceInstanceid and the IP address in RemoteAddrPtr equals the IP address in [SdClientServiceAllowedProvider](#), SD shall remove this IP address from this ClientService ACL [SdClientServiceAllowedProvider](#).] ([SRS\\_Eth\\_00165](#))

**[SWS\_Sd\_00782]{DRAFT}** [If function Sd\_AclUpdate is called with RequestType SD\_ACL\_REMOVE\_CONSUMER and the SD has a ServerService with service ID equals SdServiceId and service Instance ID equals ServiceInstanceid, SD shall remove the IP address in RemoteAddrPtr from this ServerService ACL [SdServerServiceAllowedConsumers](#).] ([SRS\\_Eth\\_00165](#))

**[SWS\_Sd\_00783]{DRAFT}** [If function Sd\_AclUpdate is called with wrong parameters e.g. RequestType SD\_ACL\_REMOVE\_PROVIDER while this IP address is not equal to the IP address in this ClientService ACL [SdClientServiceAllowedProvider](#) or there is no ClientService with this SdServiceId or this ServiceInstanceid, function shall return E\_NOT\_OK.] ([SRS\\_Eth\\_00165](#))

**[SWS\_Sd\_00784]{DRAFT}** [SD shall save the updated ACL in the NVM block referenced by this ACL SdAclCheckBlockDescriptorRef.] ([SRS\\_Eth\\_00165](#))

## 7.9.2 ACL Policy Check

This chapter describes the ACL policy decision and Enforcement.

### 7.9.2.1 Client ACL

On the ClientService side:

#### 7.9.2.1.1 Offer Service

When Client Service receives an OfferService:

**[SWS\_Sd\_00785]{DRAFT}** [SD shall check the provided IP address in the Endpoint option of this OfferService entry against this ClientService ACL [SdClientServiceAllowedProvider](#).]

- If the IP address in the Endpoint option of this OfferService entry is not on this ClientService ACL [SdClientServiceAllowedProvider](#), SD shall ignore the offer. If security event reporting is enabled ([SdEnableSecurityEventReporting](#) = true), SD shall log a security event SEV\_SOME\_IP\_ACL\_CHECK\_FAILED\_OFFER.

- If the IP address in the Endpoint option of this OfferService entry is on this ClientService ACL [SdClientServiceAllowedProvider](#), SD shall call function `SoAd_SetUniqueRemoteAddr` with remote address equal to the Endpoint option of this OfferService entry.

]([SRS\\_Eth\\_00161](#))

**[SWS\_Sd\_00786]{DRAFT}** [If the ACL check passes, SD shall call the function `SoAd_IsConnectionReady` with the parameters `SoConId` and `RemoteAddr` to check if the connection is ready and if a security association is configured that a secured connection is already established before enabling the routing.]([SRS\\_Eth\\_00161](#))

**[SWS\_Sd\_00787]{DRAFT}** [If function `SoAd_IsConnectionReady` return `TCPIP_E_NOT_OK`, SD shall ignore the offer.]([SRS\\_Eth\\_00161](#))

**[SWS\_Sd\_00788]{DRAFT}** [If the ACL check passes and the connection is ready, SD shall enable the routing of the needed methods and events from this service by calling function `SoAd_EnableSpecificRouting`.]([SRS\\_Eth\\_00161](#))

See Figure 9.11: ACL Check Sequence for Service Offer.

## 7.9.2.2 Server ACL

In the Server Service side:

### 7.9.2.2.1 SubscribeEventgroup

When Server Service receives a `SubscribeEventgroup` request:

**[SWS\_Sd\_00789]{DRAFT}** [SD shall call the function `SoAd_IsConnectionReady` with the parameters `SoConId` and `RemoteAddr` to check if the connection is ready and if a security association is configured that a secured connection is already established with the client, who has sent this `SubscribeEventgroup` request.]([SRS\\_Eth\\_00162](#))

**[SWS\_Sd\_00790]{DRAFT}** [If function `SoAd_IsConnectionReady` return `TCPIP_E_NOT_OK`, SD shall ignore the subscription request.]([SRS\\_Eth\\_00162](#))

**[SWS\_Sd\_00791]{DRAFT}** [If the connection is ready, SD shall check if the remote IP address of this client and the IP address in this `SubscribeEventgroup` endpoint option are on this EventHandler's ServerService ACL [SdServerServiceAllowedConsumers](#).

- If the remote IP address of this client or the IP address in this `SubscribeEventgroup` endpoint option is not on this EventHandler's ServerService ACL [SdServerServiceAllowedConsumers](#), SD shall ignore the subscription request. If security event reporting is enabled ([SdEnableSecurityEventReporting](#) = true), SD shall log a security event `SEV_SOME_IP_ACL_CHECK_FAILED_EVENT_SUBSCRIPTION`.



- If the remote IP address of this client and the IP address in this SubscribeEventgroup endpoint option are on this EventHandler's ServerService ACL [SdServerServiceAllowedConsumers](#), SD shall call function `SoAd_EnableSpecificRouting` to enable the routing of event updates `AddClientToFanOut`.

]([SRS\\_Eth\\_00162](#))

See Figure 9.12: ACL Check Sequence for SubscribeEventgroup.

#### 7.9.2.2.2 Method Request

When Server Service receive a Method call request:

**[SWS\_Sd\_00792]{DRAFT}** [Method routing shall not be enabled in the initialization for the ServerServices, which have ACL check enabled.]([SRS\\_Eth\\_00163](#))

**[SWS\_Sd\_00793]{DRAFT}** [If function `Sd_AclPolicyCheck` is called by SoAd module, SD shall trigger ACL policy check for this ServerService.]([SRS\\_Eth\\_00163](#))

Note: In SoAd, if the routing of the received method call request is not enabled and the ACL check is enabled (`SoAdRxSomeIpAclCheck` equals `TRUE`) for the selected socket route, SoAd will call the function `Sd_AclPolicyCheck` to trigger ACL policy check in the SD module.

**[SWS\_Sd\_00794]{DRAFT}** [SD shall call the function `SoAd_IsConnectionReady` with the parameters `SoConId` and `RemoteAddr` to check if the connection is ready and if a security association is configured that a secured connection is already established with the client, who has sent this Method call request.]([SRS\\_Eth\\_00163](#))

**[SWS\_Sd\_00795]{DRAFT}** [If function `SoAd_IsConnectionReady` return `TCPIP_E_NOT_OK`, SD shall ignore the Method call request and function `Sd_AclPolicyCheck` shall return `E_NOT_OK`.]([SRS\\_Eth\\_00163](#))

**[SWS\_Sd\_00796]{DRAFT}** [If the connection is ready, SD shall check if the remote IP address of this client is on this Method's ServerService ACL [SdServerServiceAllowedConsumers](#).

- If the remote IP address of this client is not on this Method's ServerService ACL [SdServerServiceAllowedConsumers](#), SD shall ignore the Method call request and function `Sd_AclPolicyCheck` shall return `E_NOT_OK`. If security event reporting is enabled (`SdEnableSecurityEventReporting` = `true`), SD shall log a security event `SEV_SOME_IP_ACL_CHECK_FAILED_METHOD_REQUEST`.
- If the remote IP address of this client is on this Method's ServerService ACL [SdServerServiceAllowedConsumers](#), SD shall call function `SoAd_EnableSpecificRouting` to enable the Method routing and function `Sd_AclPolicyCheck` shall return `E_OK`.

]([SRS\\_Eth\\_00163](#))



See Figure 9.13: ACL Check Sequence for Method Call Request.

## 7.10 Security Events

**[SWS\_Sd\_00797]{DRAFT}** [If security event reporting has been enabled for the SD module ([SdEnableSecurityEventReporting](#) = true) the respective security events shall be reported to the IdsM via the interfaces defined in AUTOSAR\_SWS\_BSWGeneral [2].] ([SRS\\_Eth\\_00166](#))

**[SWS\_Sd\_00798]** [A SdClientService shall only connect to one offered service instance of the same service at the same time. When a client is already connected to a service instance it shall ignore all Offer Service entries for the same service instance with different endpoint as long as the already connected service instance is considered available and report SEV\_DUPLICATE\_OFFER.] ()

Note: Service not considered available with e.g. StopOffer received, TTL expired or reboot detected.

Note: To ensure that this scenario cannot be exploited, SOME/IP-ACL can be used.

**[SWS\_Sd\_00114] Security events for Service Discovery (Sd)** [

Name	Description	ID
SEV_SOME_IP_ACL_CHECK_FAILED_OFFER	ACL check for a service offer failed	84
SEV_SOME_IP_ACL_CHECK_FAILED_EVENT_SUBSCRIPTION	ACL check for a subscribe event group request failed	85
SEV_SOME_IP_ACL_CHECK_FAILED_METHOD_REQUEST	ACL check for a method request failed	86
SEV_DUPLICATE_OFFER	SD rejected Offer for a ServiceInstance which is already offered by a different endpoint and TTL still valid.	88

] ([RS\\_Ids\\_00810](#))

**Note:** Context data (Remote IP address of communication partner, Service ID) should be saved to those security events.

## 7.11 Error Classification

Section 7.2 "Error Handling" of the document "General Specification of Basic Software Modules" [2, SWS BSW General] describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

### 7.11.1 Development Errors

#### [SWS\_Sd\_00107] Definiton of development errors in module Sd [

Type of error	Related error code	Error value
SD has not been initialized	SD_E_UNINIT	0x01
Null pointer has been passed as an argument	SD_E_PARAM_POINTER	0x02
Invalid mode request	SD_E_INV_MODE	0x03
Invalid Id	SD_E_INV_ID	0x04
Initialization failed	SD_E_INIT_FAILED	0x05

]()

**[SWS\_Sd\_00108]** [The detection of development errors shall be configurable (ON / OFF) at pre-compile time. The switch SdDevErrorDetect (see chapter 9) shall activate or deactivate the detection of all development errors.]()

**[SWS\_Sd\_00109]** [If the SdDevErrorDetect switch is enabled API parameter checking is enabled.]()

Note: The detection of production code errors cannot be switched off.

**[SWS\_Sd\_00110]** [Detected development errors shall be reported to the Det\_Report Error service of the Default Error Tracer (DET) if the pre-processor switch SdDevError Detect is set (see chapter 10).]()

### 7.11.2 Runtime Errors

#### [SWS\_Sd\_00742] Definiton of runtime errors in module Sd [

Type of error	Related error code	Error value
Retry was not successful	SD_E_COUNT_OF_RETRY_SUBSCRIPTION_EXCEEDED	0x06

]()

### 7.11.3 Transient Faults

There are no transient faults.

### 7.11.4 Production Errors

There are no Production Errors.

## 7.11.5 Extended Production Errors

### 7.11.5.1 SD\_E\_OUT\_OF\_RES

Error Name:	SD_E_OUT_OF_RES	
Short Description:	SD out of resources	
Long Description:	SD Instance does not have SoAd socket resources left to add client to Fan-Out.	
Recommended DTC:	N/A	
Detection Criteria:	FAIL	Every time when a Socket connection has to be opened but no Wildcard Socket Connection is available.
	PASS	After first startup until first error occurred.
Secondary Parameters:	Local IP-Address and Port Number of Socket Connection Group that has not enough Wildcard Socket Connections left	
Time Required:	N/A	
Monitor Frequency	Continuous	
MIL illumination:	N/A	

### 7.11.5.2 SD\_E\_MALFORMED\_MSG

Error Name:	SD_E_MALFORMED_MSG	
Short Description:	SD received malformed SOME/IP-SD message	
Long Description:	<p>The Service Discovery module received an inconsistent SOME/IP-SD message. This includes:</p> <ul style="list-style-type: none"> <li>• Inconsistent combination of SOME/IP length, entries length, and options length</li> <li>• Inconsistent length field of option</li> <li>• Illegal values of fields (e.g. IP Addresses and Ports).</li> </ul>	
Recommended DTC:	N/A	
Detection Criteria:	FAIL	Every time a malformed SOME/IP-SD message has been received
	PASS	After first startup until first error occurred.
Secondary Parameters:	IP Address of Sender (Source IP Address)	
Time Required:	N/A	
Monitor Frequency	Continuous	
MIL illumination:	N/A	

### 7.11.5.3 SD\_E\_SUBSCR\_NACK\_RECV

Error Name:	SD_E_SUBSCR_NACK_RECV	
Short Description:	SD received SubscribeEventgroupNack entry	
Long Description:	The Service Discovery module received a SubscribeEventgroupNack entry, which is not expected.	
Recommended DTC:	N/A	
Detection Criteria:	FAIL	Every time a NACK is received.
	PASS	After first startup until first error occurred.
Secondary Parameters:	IP Address of Sender (Source IP Address)	
Time Required:	N/A	
Monitor Frequency	Continuous	
MIL illumination:	N/A	

## 8 API specification

### 8.1 Imported types

[SWS\_Sd\_00117] Definition of imported datatypes of module Sd [

Module	Header File	Imported Type
ComStack_Types	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
SoAd	SoAd.h	SoAd_RoutingGroupIdType
	SoAd.h	SoAd_SoConIdType
	SoAd.h	SoAd_SoConModeType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType
Tcplp	Tcplp.h	Tcplp_DomainType
	Tcplp.h	Tcplp_IpAddrAssignmentType
	Tcplp.h	Tcplp_IpAddrStateType
	Tcplp.h	Tcplp_SockAddrType

]()

### 8.2 Type definitions

#### 8.2.1 Sd\_ConfigType

[SWS\_Sd\_00690] Definition of datatype Sd\_ConfigType [

<b>Name</b>	Sd_ConfigType	
<b>Kind</b>	Structure	
<b>Elements</b>	implementation specific	
	<b>Type</b>	–
	<b>Comment</b>	The content of the configuration data structure is implementation specific.
<b>Description</b>	Configuration data structure of Sd module.	
<b>Available via</b>	Sd.h	

]()

## 8.2.2 Sd\_ServerServiceSetStateType

### [SWS\_Sd\_00118] Definition of datatype Sd\_ServerServiceSetStateType [

<b>Name</b>	Sd_ServerServiceSetStateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_SERVER_SERVICE_ DOWN	0x00	–
	SD_SERVER_SERVICE_ AVAILABLE	0x01	–
<b>Description</b>	This type defines the Server states that are reported to the SD using the expected API Sd_ServerServiceSetState.		
<b>Available via</b>	Sd.h		

]()

## 8.2.3 Sd\_ClientServiceSetStateType

### [SWS\_Sd\_00405] Definition of datatype Sd\_ClientServiceSetStateType [

<b>Name</b>	Sd_ClientServiceSetStateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_CLIENT_SERVICE_ RELEASED	0x00	–
	SD_CLIENT_SERVICE_ REQUESTED	0x01	–
<b>Description</b>	This type defines the Client states that are reported to the BswM using the expected API Sd_ClientServiceSetState.		
<b>Available via</b>	Sd.h		

]()

## 8.2.4 Sd\_ConsumedEventGroupSetStateType

### [SWS\_Sd\_00550] Definition of datatype Sd\_ConsumedEventGroupSetStateType [

<b>Name</b>	Sd_ConsumedEventGroupSetStateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_CONSUMED_ EVENTGROUP_ RELEASED	0x00	–
	SD_CONSUMED_ EVENTGROUP_ REQUESTED	0x01	–
<b>Description</b>	This type defines the subscription policy by consumed EventGroup for the Client Service.		
<b>Available via</b>	Sd.h		

]()

## 8.2.5 Sd\_ClientServiceCurrentStateType

[SWS\_Sd\_00551] Definition of datatype Sd\_ClientServiceCurrentStateType [

<b>Name</b>	Sd_ClientServiceCurrentStateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_CLIENT_SERVICE_DOWN	0x00	–
	SD_CLIENT_SERVICE_AVAILABLE	0x01	–
<b>Description</b>	This type defines the modes to indicate the current mode request of a Client Service.		
<b>Available via</b>	Sd.h		

]()

## 8.2.6 Sd\_ConsumedEventGroupCurrentStateType

[SWS\_Sd\_00552] Definition of datatype Sd\_ConsumedEventGroupCurrentStateType [

<b>Name</b>	Sd_ConsumedEventGroupCurrentStateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_CONSUMED_EVENTGROUP_DOWN	0x00	–
	SD_CONSUMED_EVENTGROUP_AVAILABLE	0x01	–
<b>Description</b>	This type defines the subscription policy by consumed EventGroup for the Client Service.		
<b>Available via</b>	Sd.h		

]()

## 8.2.7 Sd\_EventHandlerCurrentStateType

[SWS\_Sd\_00553] Definition of datatype Sd\_EventHandlerCurrentStateType [

<b>Name</b>	Sd_EventHandlerCurrentStateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_EVENT_HANDLER_RELEASED	0x00	–
	SD_EVENT_HANDLER_REQUESTED	0x01	–
<b>Description</b>	This type defines the subscription policy by EventHandler for the Server Service.		
<b>Available via</b>	Sd.h		

]()

## 8.2.8 Sd\_ConfigOptionStringType

[SWS\_Sd\_91002] Definition of datatype Sd\_ConfigOptionStringType [

<b>Name</b>	Sd_ConfigOptionStringType
<b>Kind</b>	Const Pointer
<b>Type</b>	const uint8*
<b>Description</b>	Type for a zero-terminated string of configuration options.
<b>Available via</b>	Sd.h

]()

## 8.2.9 Sd\_ServiceGroupIdType

[SWS\_Sd\_91008] Definition of datatype Sd\_ServiceGroupIdType [

<b>Name</b>	Sd_ServiceGroupIdType		
<b>Kind</b>	Type		
<b>Derived from</b>	uint16		
<b>Range</b>	0..65535	–	Zero-based integer number
<b>Description</b>	The AUTOSAR ServiceDiscovery module's SdServiceGroup object identifier.		
<b>Available via</b>	Sd.h		

]()

## 8.2.10 Sd\_ServiceAclUpdateType

[SWS\_Sd\_91009]{DRAFT} Definition of datatype Sd\_AclUpdateType [

<b>Name</b>	Sd_AclUpdateType (draft)		
<b>Kind</b>	Enumeration		
<b>Range</b>	SD_ACL_ADD_PROVIDER	0x00	Add this IP address to providers ACL sdClient ServiceAllowedProviders
	SD_ACL_ADD_CONSUMER	0x01	Add this IP address to consumers ACL sd ServerServiceAllowedConsumers
	SD_ACL_REMOVE_PROVIDER	0x02	Remove this IP address from providers ACL sdClientServiceAllowedProviders
	SD_ACL_REMOVE_CONSUMER	0x03	Remove this IP address from consumers ACL sdServerServiceAllowedConsumers
<b>Description</b>	This type defines the required ACL update action. <b>Tags:</b> atp.Status=draft		
<b>Available via</b>	Sd.h		

]([SRS\\_Eth\\_00165](#))



## 8.3 Function definitions

This is a list of functions provided for upper layer modules.

### 8.3.1 Sd\_Init

#### [SWS\_Sd\_00119] Definition of API function Sd\_Init [

<b>Service Name</b>	Sd_Init	
<b>Syntax</b>	<pre>void Sd_Init (     const Sd_ConfigType* ConfigPtr )</pre>	
<b>Service ID [hex]</b>	0x01	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	ConfigPtr	Pointer to a selected configuration structure.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Initializes the Service Discovery.	
<b>Available via</b>	Sd.h	

]()

[SWS\_Sd\_00120] [The Sd\_Init function shall initialize the state machines for all Service Instances according to SWS\_SD\_00020 and SWS\_SD\_00021.]()

[SWS\_Sd\_00121] [The Sd\_Init function shall internally store the configuration data address to enable subsequent API calls to access the configuration data.]()

[SWS\_Sd\_00122] [The Sd\_Init function shall remember internally the successful initialization for other API functions to check for proper module initialization.]()

### 8.3.2 Sd\_GetVersionInfo

#### [SWS\_Sd\_00124] Definition of API function Sd\_GetVersionInfo [

<b>Service Name</b>	Sd_GetVersionInfo	
<b>Syntax</b>	<pre>void Sd_GetVersionInfo (     Std_VersionInfoType* versioninfo )</pre>	
<b>Service ID [hex]</b>	0x02	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	





<b>Parameters (out)</b>	versioninfo	Pointer to where to store the version information of this module.
<b>Return value</b>	None	
<b>Description</b>	Returns the version information of this module.	
<b>Available via</b>	Sd.h	

]()

**[SWS\_Sd\_00125]** [The Sd\_GetVersionInfo function shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers]

**[SWS\_Sd\_00126]** [Configuration of Sd\_GetVersionInfo: This function shall be pre compile time configurable On/Off by the configuration parameter: SdVersionInfoApi]  
( )

**[SWS\_Sd\_00497]** [If development error detection for the Service Discovery module is enabled, then the function Sd\_GetVersionInfo shall check whether the parameter VersioninfoPtr is a NULL pointer (NULL\_PTR). If VersioninfoPtr is a NULL pointer, then the function Sd\_GetVersionInfo shall raise the development error SD\_E\_PARAM\_POINTER and return.] ( )

### 8.3.3 Sd\_ServerServiceSetState

**[SWS\_Sd\_00496]** Definition of API function Sd\_ServerServiceSetState [

<b>Service Name</b>	Sd_ServerServiceSetState	
<b>Syntax</b>	Std_ReturnType Sd_ServerServiceSetState ( uint16 SdServerServiceHandleId, Sd_ServerServiceSetStateType ServerServiceState )	
<b>Service ID [hex]</b>	0x07	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	SdServerServiceHandleId	ID to identify the Server Service Instance.
	ServerServiceState	The state the Server Service Instance shall be set to.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: State accepted E_NOT_OK: State not accepted
<b>Description</b>	This API function is used by the BswM to set the Server Service Instance state.	
<b>Available via</b>	Sd.h	

]()

**[SWS\_Sd\_00407]** [If development error detection is enabled and the Service Discovery module has not been initialized using `Sd_Init()`, the `Sd_ServerServiceSetState` function shall raise the development error code `SD_E_UNINIT` and the `Sd_ServerServiceSetState` function shall return `E_NOT_OK`.]()

**[SWS\_Sd\_00408]** [If the parameter `ServerServiceState` has an undefined value, the Service Discovery module shall not store the requested mode and return `E_NOT_OK`.

In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code `SD_E_INV_MODE`.]()

**[SWS\_Sd\_00607]** [If the parameter `SdServerServiceHandleId` has an invalid value, the Service Discovery Module shall not store the requested mode and return `E_NOT_OK`. In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code `SD_E_INV_ID`.]()

### 8.3.4 Sd\_ClientServiceSetState

**[SWS\_Sd\_00409]** Definition of API function `Sd_ClientServiceSetState` [

<b>Service Name</b>	Sd_ClientServiceSetState	
<b>Syntax</b>	<pre>Std_ReturnType Sd_ClientServiceSetState (     uint16 ClientServiceHandleId,     Sd_ClientServiceSetStateType ClientServiceState )</pre>	
<b>Service ID [hex]</b>	0x08	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	ClientServiceHandleId	ID to identify the Client Service Instance.
	ClientServiceState	The state the Client Service Instance shall be set to.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: State accepted
		E_NOT_OK: State not accepted
<b>Description</b>	This API function is used by the BswM to set the Client Service Instance state.	
<b>Available via</b>	Sd.h	

]()

**[SWS\_Sd\_00410]** [If development error detection is enabled and the Service Discovery module has not been initialized using `Sd_Init()`, the `Sd_ClientServiceSetState` function shall raise the development error code `SD_E_UNINIT` and the `Sd_ClientServiceSetState` function shall return `E_NOT_OK`.]()

**[SWS\_Sd\_00411]** [If the parameter `ClientServiceState` has an undefined value, the Service Discovery module shall not store the requested mode and return `E_NOT_OK`.

In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code `SD_E_INV_MODE`.]()

**[SWS\_Sd\_00608]** [If the parameter ClientServiceHandleId has an invalid value, the Service Discovery module shall not store the requested mode and return E\_NOT\_OK. In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code SD\_E\_INV\_ID.]()

### 8.3.5 Sd\_ConsumedEventGroupSetState

**[SWS\_Sd\_00560]** Definition of API function Sd\_ConsumedEventGroupSetState [

<b>Service Name</b>	Sd_ConsumedEventGroupSetState	
<b>Syntax</b>	<pre>Std_ReturnType Sd_ConsumedEventGroupSetState (     uint16 SdConsumedEventGroupHandleId,     Sd_ConsumedEventGroupSetStateType ConsumedEventGroupState )</pre>	
<b>Service ID [hex]</b>	0x09	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	SdConsumedEventGroupHandleId	ID to identify the Consumed Eventgroup
	ConsumedEventGroupState	The state the EventGroup shall be set to.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: State accepted E_NOT_OK: State not accepted
<b>Description</b>	This API function is used by the BswM to set the requested state of the EventGroupStatus.	
<b>Available via</b>	Sd.h	

]()

**[SWS\_Sd\_00469]** [If development error detection is enabled and the Service Discovery module has not been initialized using Sd\_Init(), the Sd\_ConsumedEventGroupSetState function shall raise the development error code SD\_E\_UNINIT and the Sd\_ConsumedEventGroupSetState function shall return E\_NOT\_OK.]()

**[SWS\_Sd\_00470]** [If ConsumedEventGroupSetState has an undefined value, the Service Discovery module shall not store the requested mode and return E\_NOT\_OK.

In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code SD\_E\_INV\_MODE.]()

**[SWS\_Sd\_00609]** [If the parameter SdConsumedEventGroupHandleId has an invalid value, the Service Discovery module shall not store the requested mode and return E\_NOT\_OK. In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code SD\_E\_INV\_ID.]()

### 8.3.6 Sd\_LocalIpAddrAssignmentChg

#### [SWS\_Sd\_00412] Definition of API function Sd\_LocalIpAddrAssignmentChg [

<b>Service Name</b>	Sd_LocalIpAddrAssignmentChg	
<b>Syntax</b>	<pre>void Sd_LocalIpAddrAssignmentChg (     SoAd_SoConIdType SoConId,     TcpIp_IpAddrStateType State )</pre>	
<b>Service ID [hex]</b>	0x05	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different SoConIds. Non Reentrant for the same SoConId.	
<b>Parameters (in)</b>	SoConId	socket connection index specifying the socket connection where the IP address assignment has changed.
	State	state of IP address assignment.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This function gets called by the SoAd if an IP address assignment related to a socket connection changes (i.e. new address assigned or assigned address becomes invalid).	
<b>Available via</b>	Sd.h	

]()

**[SWS\_Sd\_00471]** [If development error detection is enabled and the Service Discovery module has not been initialized using Sd\_Init(), the Sd\_LocalIpAddrAssignmentChg function shall raise the development error code SD\_E\_UNINIT and the Sd\_LocalIpAddrAssignmentChg function shall return without further action.]()

**[SWS\_Sd\_00472]** [If the parameter State has an undefined value, the Service Discovery module shall not store the requested mode and return.

In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code SD\_E\_INV\_MODE.]()

**[SWS\_Sd\_00610]** [If the parameter SoConId has an invalid value, the Service Discovery module shall not store the requested mode and return. In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code SD\_E\_INV\_ID.]()

### 8.3.7 Sd\_SoConModeChg

#### [SWS\_Sd\_91003] Definition of callback function Sd\_SoConModeChg [

<b>Service Name</b>	Sd_SoConModeChg	
<b>Syntax</b>	<pre>void Sd_SoConModeChg (     SoAd_SoConIdType SoConId,     SoAd_SoConModeType Mode )</pre>	





<b>Service ID [hex]</b>	0x43	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different SoConIds. Non reentrant for the same SoConId.	
<b>Parameters (in)</b>	SoConId	socket connection index specifying the socket connection with the mode change.
	Mode	new mode
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Notification about a SoAd socket connection state change, e.g. socket connection gets online	
<b>Available via</b>	Sd.h	

|()

### 8.3.8 Sd\_ServiceGroupStart

#### [SWS\_Sd\_91006] Definition of API function Sd\_ServiceGroupStart [

<b>Service Name</b>	Sd_ServiceGroupStart	
<b>Syntax</b>	<pre>void Sd_ServiceGroupStart (     Sd_ServiceGroupIdType ServiceGroupId )</pre>	
<b>Service ID [hex]</b>	0x44	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different SdServiceGroupS. Non reentrant for the same SdServiceGroup.	
<b>Parameters (in)</b>	ServiceGroupId	Id of SdServiceGroup to be started
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Starts a preconfigured SdServiceGroup. For example, OfferService entries will be sent out after the call of Sd_ServiceGroupStart() for all ServerServices of a SdServiceGroup, which are not requested yet.	
<b>Available via</b>	Sd.h	

|()

### 8.3.9 Sd\_ServiceGroupStop

#### [SWS\_Sd\_91007] Definition of API function Sd\_ServiceGroupStop [

<b>Service Name</b>	Sd_ServiceGroupStop	
<b>Syntax</b>	<pre>void Sd_ServiceGroupStop (     Sd_ServiceGroupIdType ServiceGroupId )</pre>	
<b>Service ID [hex]</b>	0x45	





<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different SdServiceGroupS. Non reentrant for the same SdServiceGroup.	
<b>Parameters (in)</b>	ServiceGroupId	Id of SdServiceGroup to be stopped
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Stops a preconfigured SdServiceGroup. For example, StopOfferService entries will be sent out after the call of Sd_ServiceGroupStop() for all ServerServices of a SdServiceGroup, which are not requested by another SdServiceGroup.	
<b>Available via</b>	Sd.h	

]()

### 8.3.10 Sd\_AclUpdate

#### [SWS\_Sd\_91010]{DRAFT} Definition of API function Sd\_AclUpdate [

<b>Service Name</b>	Sd_AclUpdate (draft)	
<b>Syntax</b>	<pre>Std_ReturnType Sd_AclUpdate (     uint16 SdServiceId,     uint16 ServiceInstanceId,     const TcpIp_SockAddrType* RemoteAddrPtr,     Sd_AclUpdateType RequestType )</pre>	
<b>Service ID [hex]</b>	0x46	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different Servicelds. Non reentrant for the same Serviceld.	
<b>Parameters (in)</b>	SdServiceld	The service ID.
	ServiceInstanceId	The service instance ID.
	RemoteAddrPtr	The IP address to be added to or removed from this service ACL.
	RequestType	The type of the update request (add or remove from providers or consumers ACL).
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: ACL has been updated. E_NOT_OK: ACL update failed
<b>Description</b>	Update Service ACL SdClientServiceAllowedProviders or SdServerServiceAllowedConsumers depending on Client or Server service by adding or removing this IP address. <b>Tags:</b> atp.Status=draft	
<b>Available via</b>	Sd.h	

] ([SRS\\_Eth\\_00165](#))

### 8.3.11 Sd\_AclPolicyCheck

#### [SWS\_Sd\_91011]{DRAFT} Definition of API function Sd\_AclPolicyCheck [

<b>Service Name</b>	Sd_AclPolicyCheck (draft)	
<b>Syntax</b>	<pre>Std_ReturnType Sd_AclPolicyCheck (     uint32 PduHeaderID,     SoAd_SoConIdType SoConId )</pre>	
<b>Service ID [hex]</b>	0x47	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	PduHeaderID	a combination of service ID and method ID
	SoConId	socket connection index specifying the socket connection on which the PDUs has been received
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: ACL check passed E_NOT_OK: ACL check failed
	<b>Description</b> Callback function, which will be provided to SoAd to be able to trigger the ACL policy check on the received Method call request <b>Tags:</b> atp.Status=draft	
<b>Available via</b>	Sd.h	

]([SRS\\_Eth\\_00163](#))

## 8.4 Callback notifications

This is a list of functions provided for other modules.

### 8.4.1 Sd\_RxIndication

#### [SWS\_Sd\_00129] Definition of callback function Sd\_RxIndication [

<b>Service Name</b>	Sd_RxIndication	
<b>Syntax</b>	<pre>void Sd_RxIndication (     PduIdType RxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID [hex]</b>	0x42	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different PduIds. Non reentrant for the same PduId.	
<b>Parameters (in)</b>	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	







<b>Return value</b>	None
<b>Description</b>	Indication of a received PDU from a lower layer communication interface module.
<b>Available via</b>	Sd.h

]()

**[SWS\_Sd\_00473]** [If development error detection is enabled and the Service Discovery module has not been initialized using Sd\_Init(), the Sd\_RxIndication function shall raise the development error code SD\_E\_UNINIT and the Sd\_RxIndication function shall return without further action.]()

**[SWS\_Sd\_00474]** [If RxPduId has an undefined value, the Service Discovery module shall discard the message and return without further action.

In case development error detection is enabled, the Service Discovery module shall additionally raise the development error code SD\_E\_INV\_ID.]()

**[SWS\_Sd\_00475]** [If development error detection is enabled: The function shall check parameter PduInfoPtr for being a null pointer. In this case, the function shall raise the development error SD\_E\_PARAM\_POINTER and return without further action.]()

## 8.5 Scheduled functions

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

### 8.5.1 Sd\_MainFunction

**[SWS\_Sd\_00130]** Definition of scheduled function Sd\_MainFunction [

<b>Service Name</b>	Sd_MainFunction
<b>Syntax</b>	void Sd_MainFunction ( void )
<b>Service ID [hex]</b>	0x06
<b>Description</b>	–
<b>Available via</b>	SchM_Sd.h

]()

**[SWS\_Sd\_00131]** [The Sd\_MainFunction shall update all counters, timers, states and phases and process the Rx and Tx data path.]()

## 8.6 Expected interfaces

In this chapter, all interfaces required from other modules are listed.

### 8.6.1 Mandatory Interfaces

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

#### [SWS\_Sd\_00133] Definition of mandatory interfaces in module Sd [

API Function	Header File	Description
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
SoAd_DisableSpecificRouting	SoAd.h	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 SoCon Id.
SoAd_EnableSpecificRouting	SoAd.h	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 SoCon Id.
SoAd_GetLocalAddr	SoAd.h	Retrieves the local address (IP address and port) actually used for the SoAd socket connection specified by SoConId, the netmask and default router
SoAd_GetPhysAddr	SoAd.h	Retrieves the physical source address of the EthIf controller used by the SoAd socket connection specified by SoConId.
SoAd_GetRemoteAddr	SoAd.h	Retrieves the remote address (IP address and port) actually used for the SoAd socket connection specified by SoConId
SoAd_GetSoConMode	SoAd.h	Returns current state of the socket connection specified by SoConId.
SoAd_IfSpecificRoutingGroupTransmit	SoAd.h	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.
SoAd_IfTransmit	SoAd.h	Requests transmission of a PDU.
SoAd_ReleaseRemoteAddr	SoAd.h	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.
SoAd_SetRemoteAddr	SoAd.h	By this API service the remote address (IP address and port) of the specified socket connection shall be set.

]()

## 8.6.2 Optional Interfaces

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

### [SWS\_Sd\_00134] Definition of optional interfaces in module Sd

API Function	Header File	Description
BswM_Sd_ClientServiceCurrentState	BswM_Sd.h	Function called by Service Discovery to indicate current state of the Client Service (available/down).
BswM_Sd_ConsumedEventGroupCurrentState	BswM_Sd.h	Function called by Service Discovery to indicate current status of the Consumed Eventgroup (available/down).
BswM_Sd_EventHandlerCurrentState	BswM_Sd.h	Function called by Service Discovery to indicate current status of the EventHandler (requested/released).
Det_ReportError	Det.h	Service to report development errors.
SoAd_CloseSoCon	SoAd.h	This service closes the socket connection specified by SoConId.
SoAd_DisableRouting	SoAd.h	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.
SoAd_EnableRouting	SoAd.h	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.
SoAd_GetSoConId	SoAd.h	Returns socket connection index related to the specified TxPduld.
SoAd_IfRoutingGroupTransmit	SoAd.h	Triggers the transmission of all If-TxPDUs identified by the parameter id after requesting the data from the related upper layer.
SoAd_OpenSoCon	SoAd.h	This service opens the socket connection specified by SoConId.
SoAd_ReleaseIpAddrAssignment	SoAd.h	By this API service the local IP address assignment used for the socket connection specified by SoConId is released.
SoAd_RequestIpAddrAssignment	SoAd.h	By this API service the local IP address assignment which shall be used for the socket connection specified by SoConId is initiated.
SoAd_SetUniqueRemoteAddr	SoAd.h	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.

]()

## 8.6.3 Configurable Interfaces

### 8.6.3.1 Sd\_CapabilityRecordMatchCallout

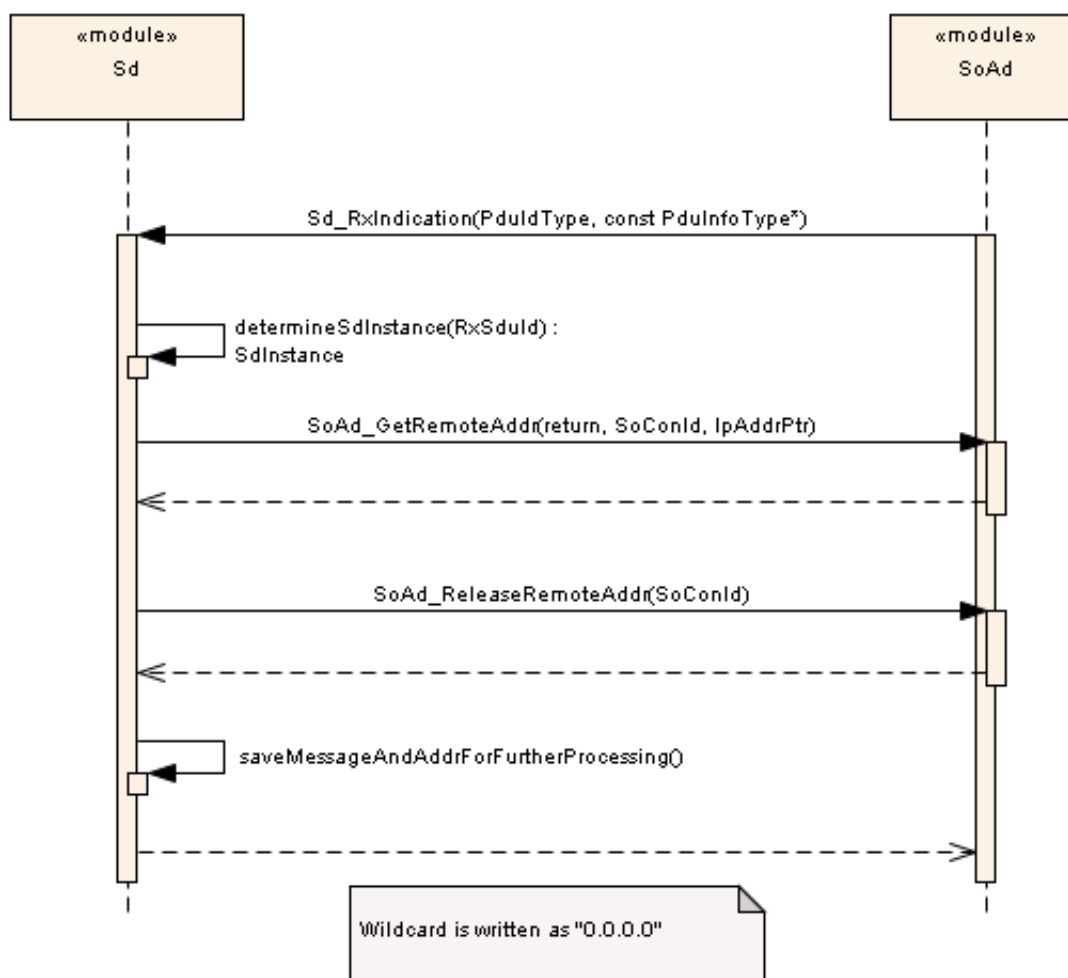
[SWS\_Sd\_91001] Definition of callout function <SdCapabilityRecordMatchCallout> [

<b>Service Name</b>	<SdCapabilityRecordMatchCallout>	
<b>Syntax</b>	<pre>boolean &lt;SdCapabilityRecordMatchCallout&gt; (     PduIdType pduID,     uint8 type,     uint16 serviceID,     uint16 instanceID,     uint8 majorVersion,     uint32 minorVersion,     const Sd_ConfigOptionStringType* receivedConfigOptionPtrArray,     const Sd_ConfigOptionStringType* configuredConfigOptionPtrArray )</pre>	
<b>Service ID [hex]</b>	0x10	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different PduIDs. Non reentrant for the same PduID.	
<b>Parameters (in)</b>	pduID	ID of the received I-PDU (used to distinguish between different SD instances)
	type	Content of the Type field of the received entry (see section 7.3.8)
	serviceID	Content of the Service ID field of the received entry (see section 7.3.8)
	instanceID	Content of the Instance ID field of the received entry (see section 7.3.8)
	majorVersion	Content of the Major Version field of the received entry (see section 7.3.8)
	minorVersion	Content of the Minor Version field of the received entry (see section 7.3.8)
	receivedConfigOptionPtrArray	NULL_PTR terminated array of pointers to zero-terminated configuration strings received in the incoming entry, i.e. received SD message (see Figure 6 - Configuration Option)
	configuredConfigOptionPtrArray	NULL_PTR terminated array of pointers to zero-terminated configuration strings configured in the local SD configuration (see Figure 6 - Configuration Option)
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	boolean	TRUE: The received configuration options match the configured ones. FALSE: The received configuration options do not match the configured ones.
<b>Description</b>	This callout is invoked to determine whether the configuration options contained in a received SD message match the ones configured in the local SD configuration (i.e., SdServerCapabilityRecord or SdClientCapabilityRecord).	
<b>Available via</b>	Sd_Externals.h	

]() This callout must be configured in the SdCapabilityRecordMatchCallout container. The name of the callout functions is given by the SdCapabilityRecordMatchCallout Name configuration element.

## 9 Sequence diagrams

### 9.1 CLIENT / SERVER: Sd\_RxIndication



**Figure 9.1: Sequence CLIENT / SERVER: Sd\_RxIndication**

## 9.2 SERVER: Response Behavior

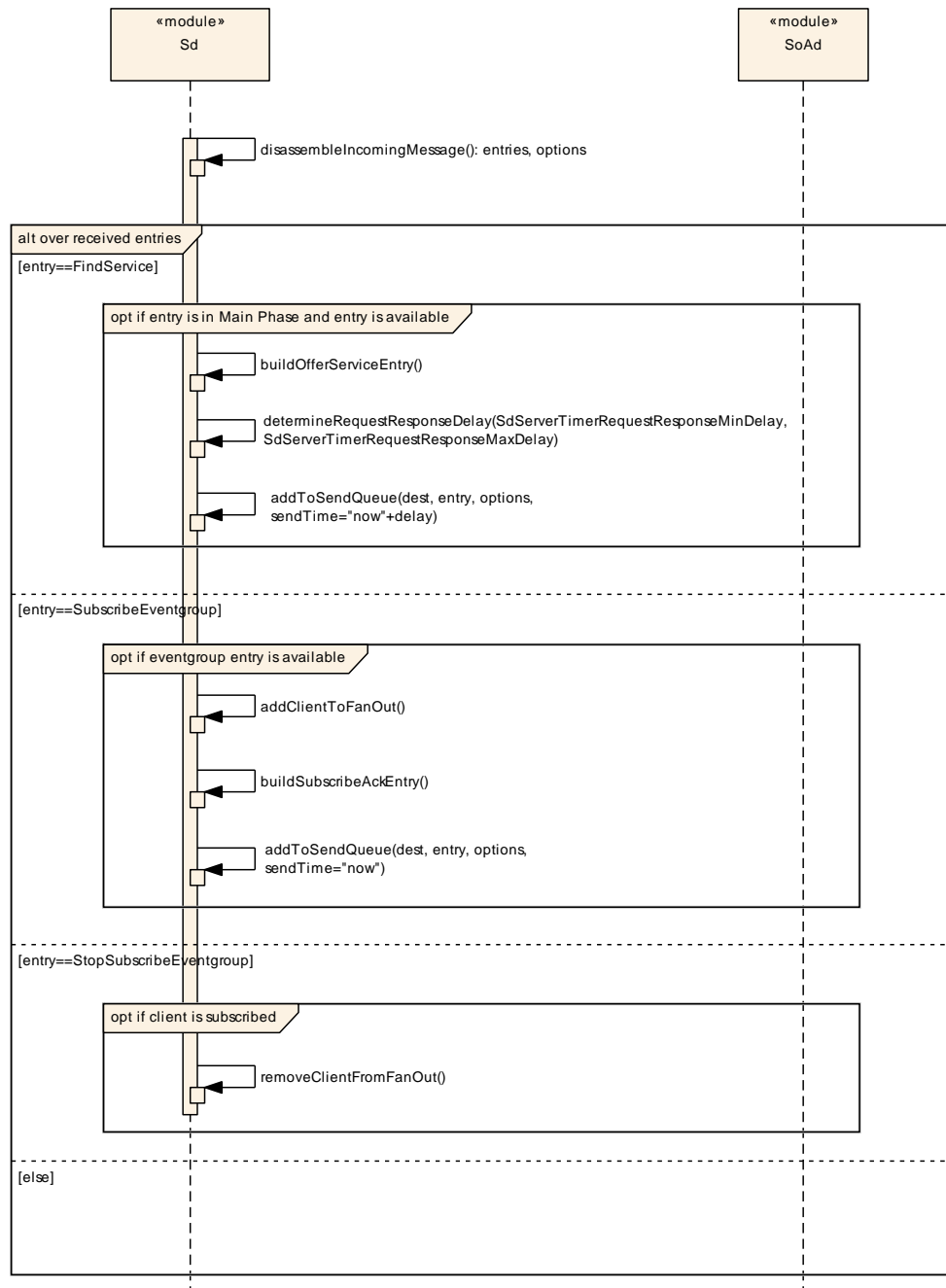


Figure 9.2: Sequence SERVER: Response Behavior

### 9.3 CLIENT: Response Behavior

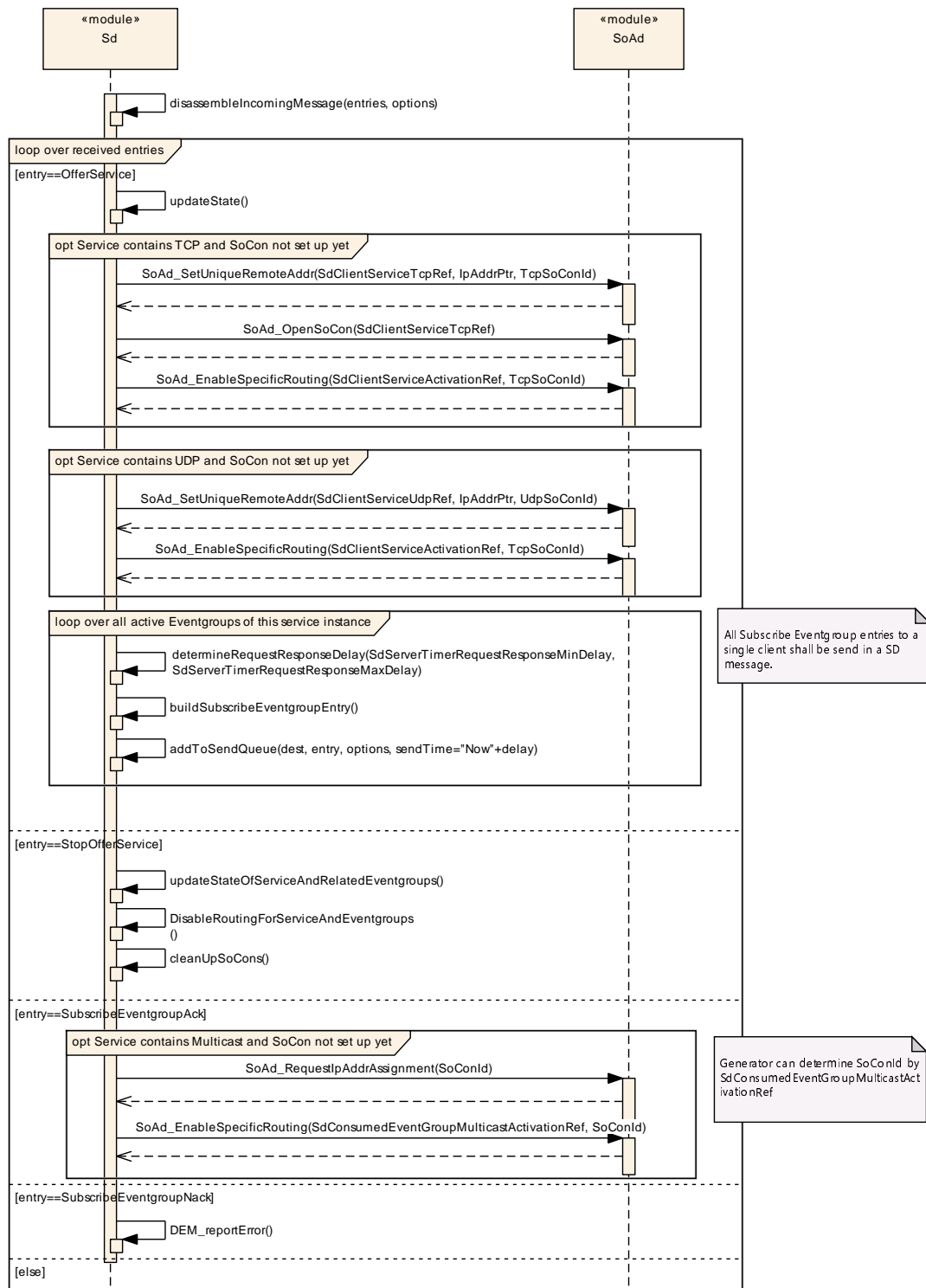
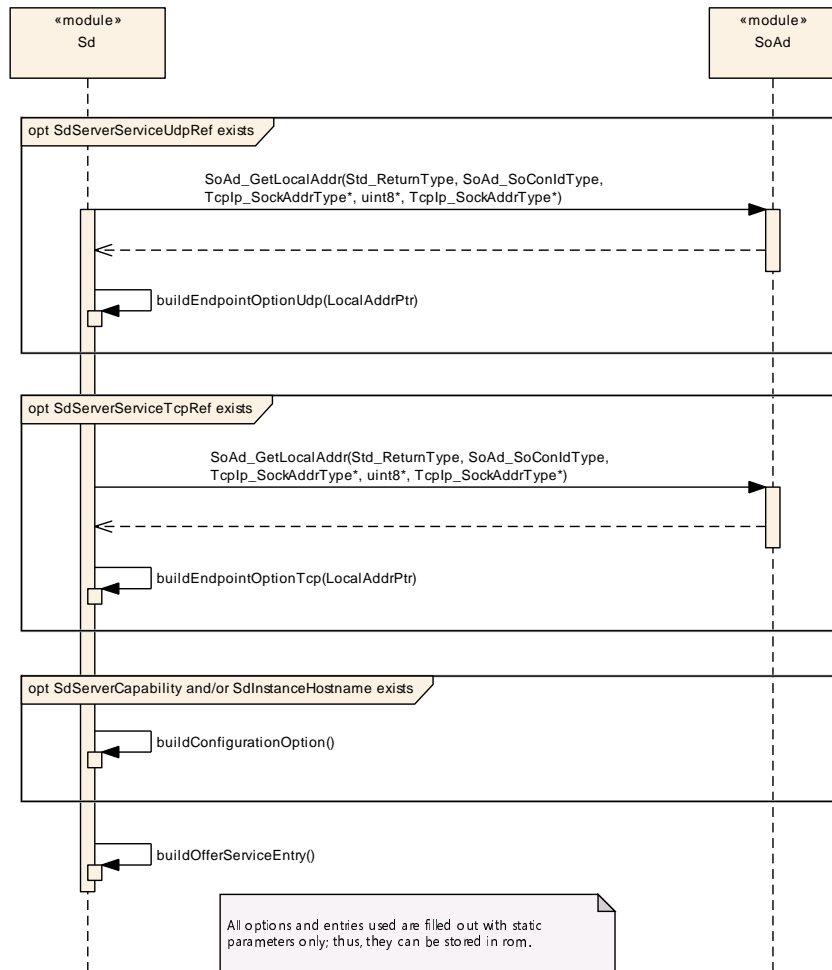


Figure 9.3: Sequence CLIENT: Response Behavior

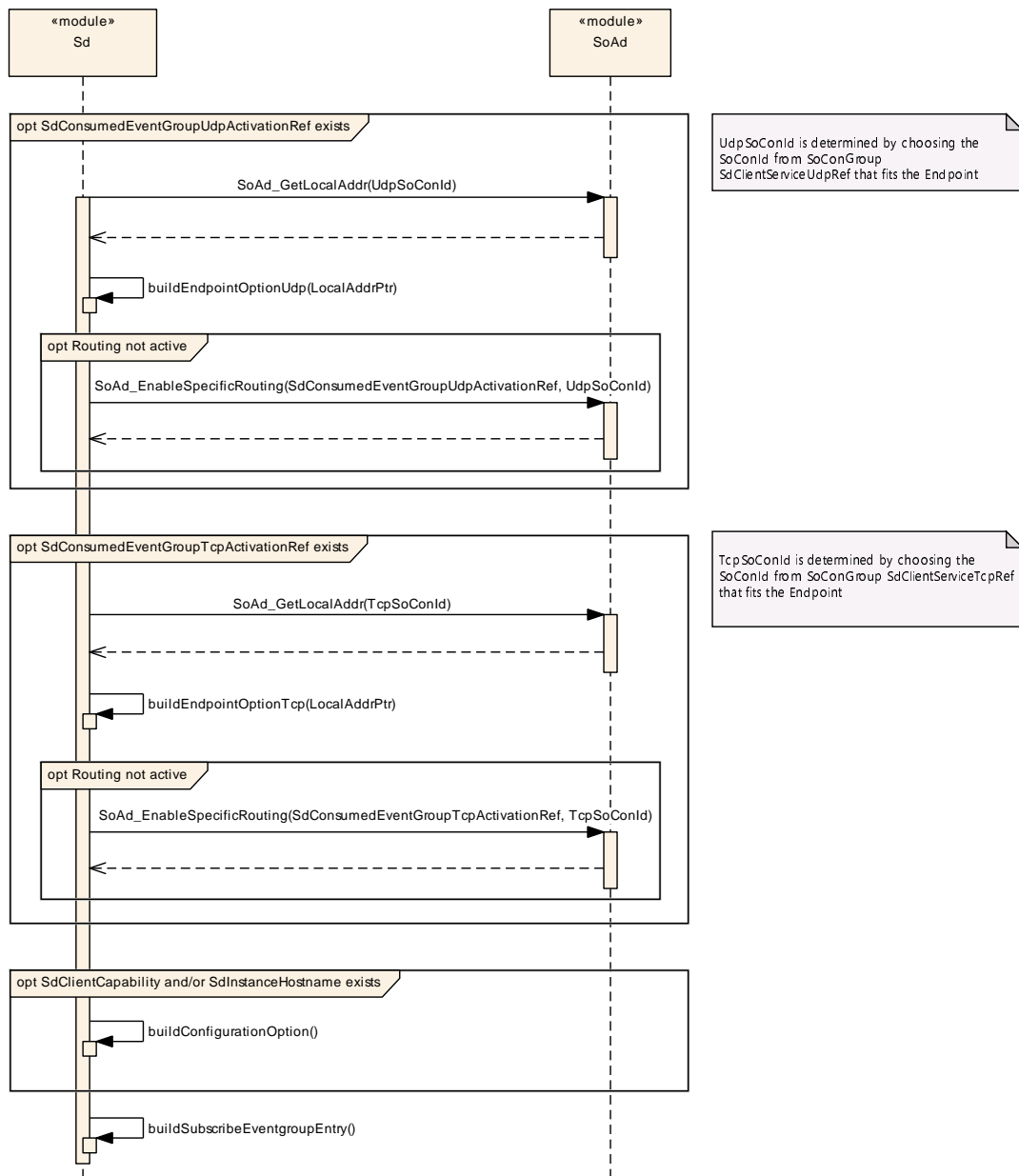
## 9.4 SERVER: buildOfferServiceEntry



**Figure 9.4: Sequence SERVER: buildOfferServiceEntry**



## 9.5 CLIENT: buildSubscribeEventgroupEntry



**Figure 9.5: Sequence CLIENT: buildSubscribeEventgroupEntry**

## 9.6 SERVER: buildSubscribeEventgroupAckEntry

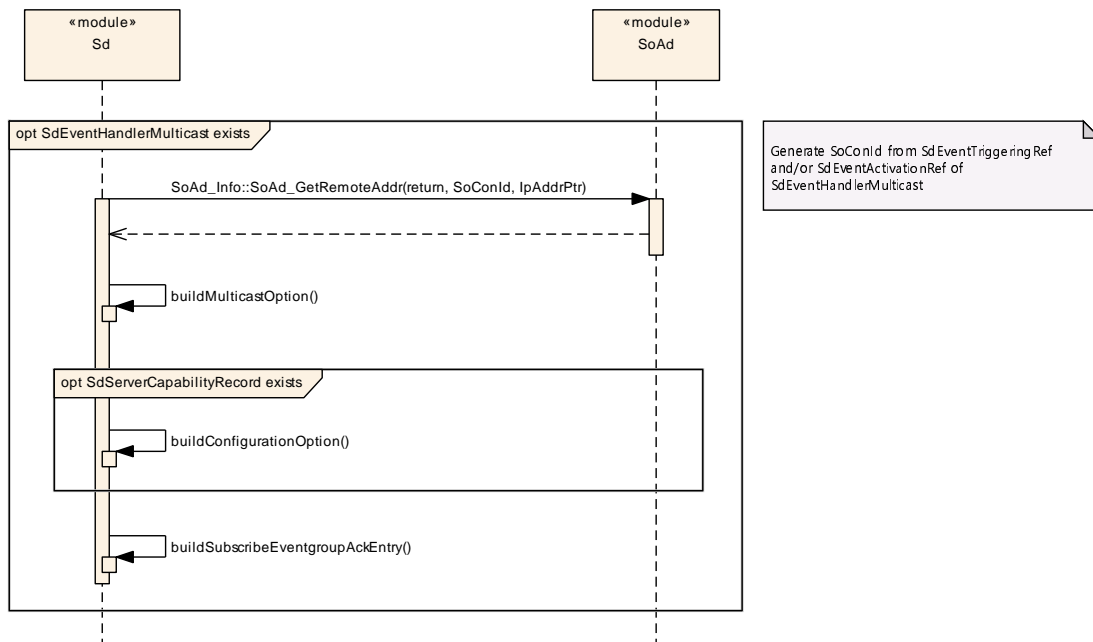


Figure 9.6: Sequence CLIENT: buildSubscribeEventgroupAckEntry

## 9.7 CLIENT / SERVER: TransmitSdMessage

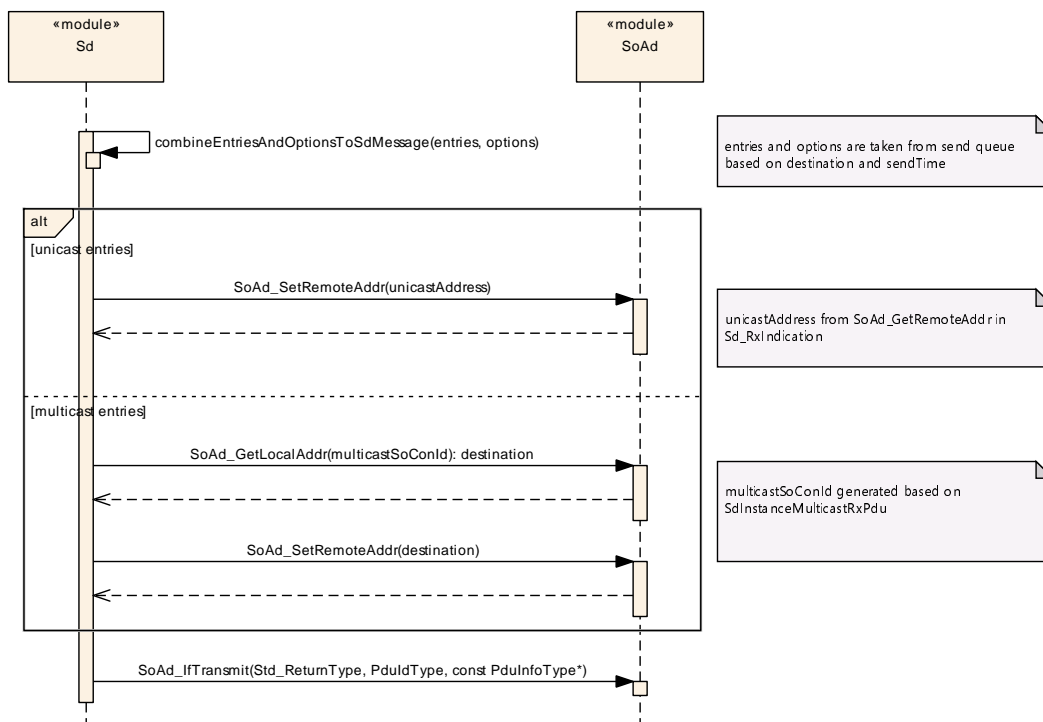


Figure 9.7: Sequence CLIENT / SERVER: TransmitSdMessage

## 9.8 SERVER: AddClientToFanOut

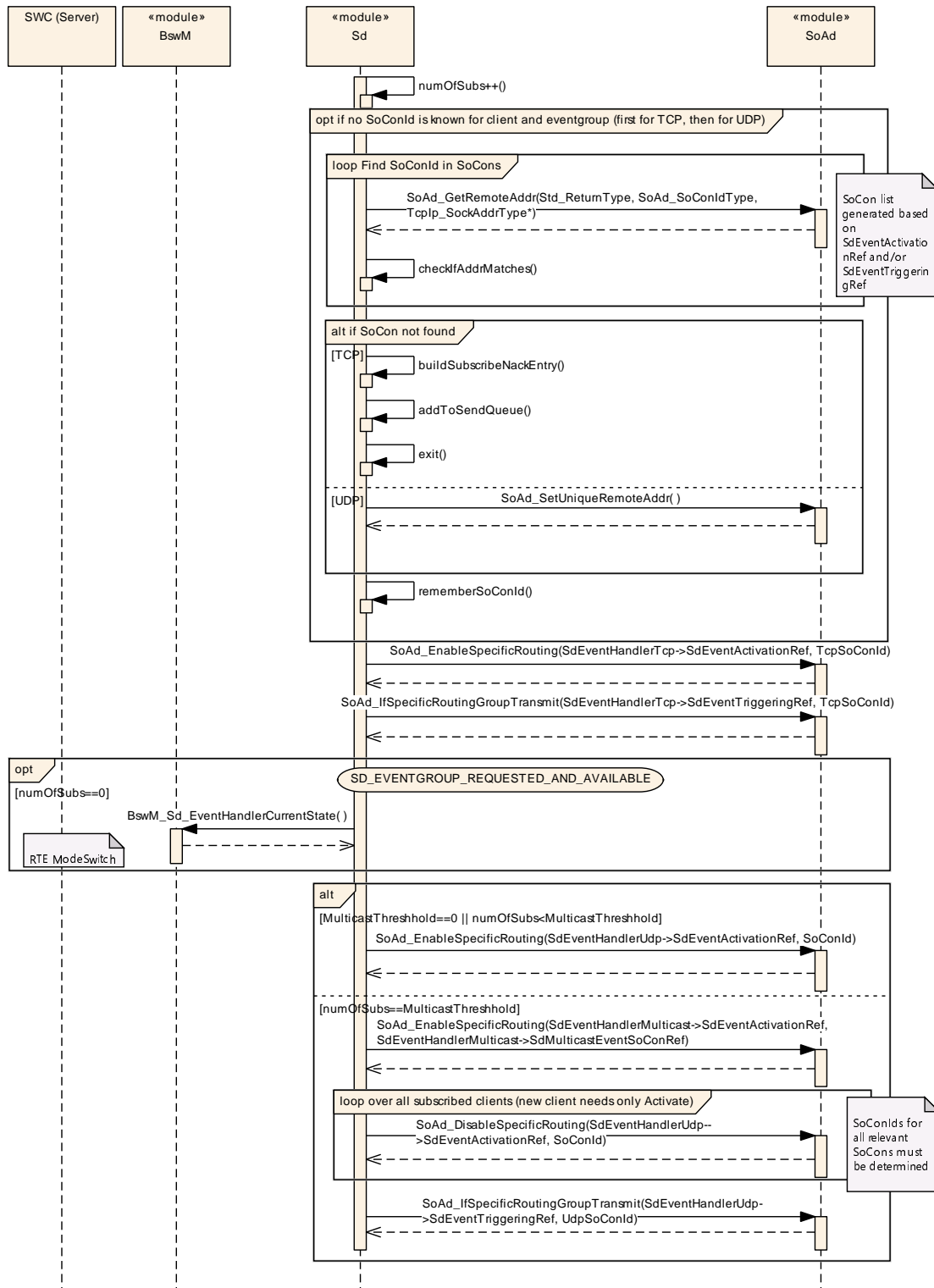


Figure 9.8: Sequence SERVER: AddClientToFanOut

## 9.9 SERVER: Start

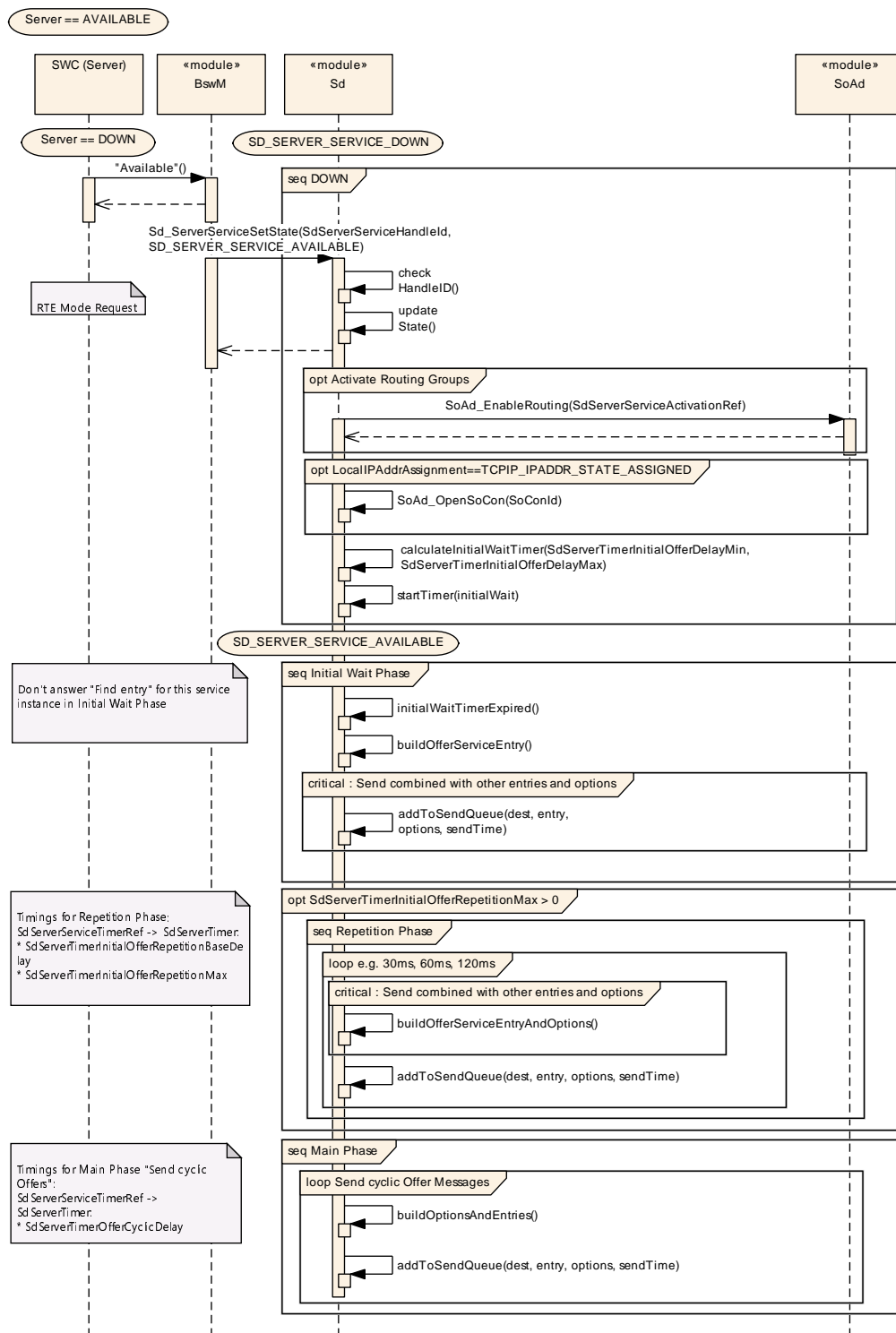


Figure 9.9: Sequence configuration variants SERVER: Start

## 9.10 CLIENT: Start

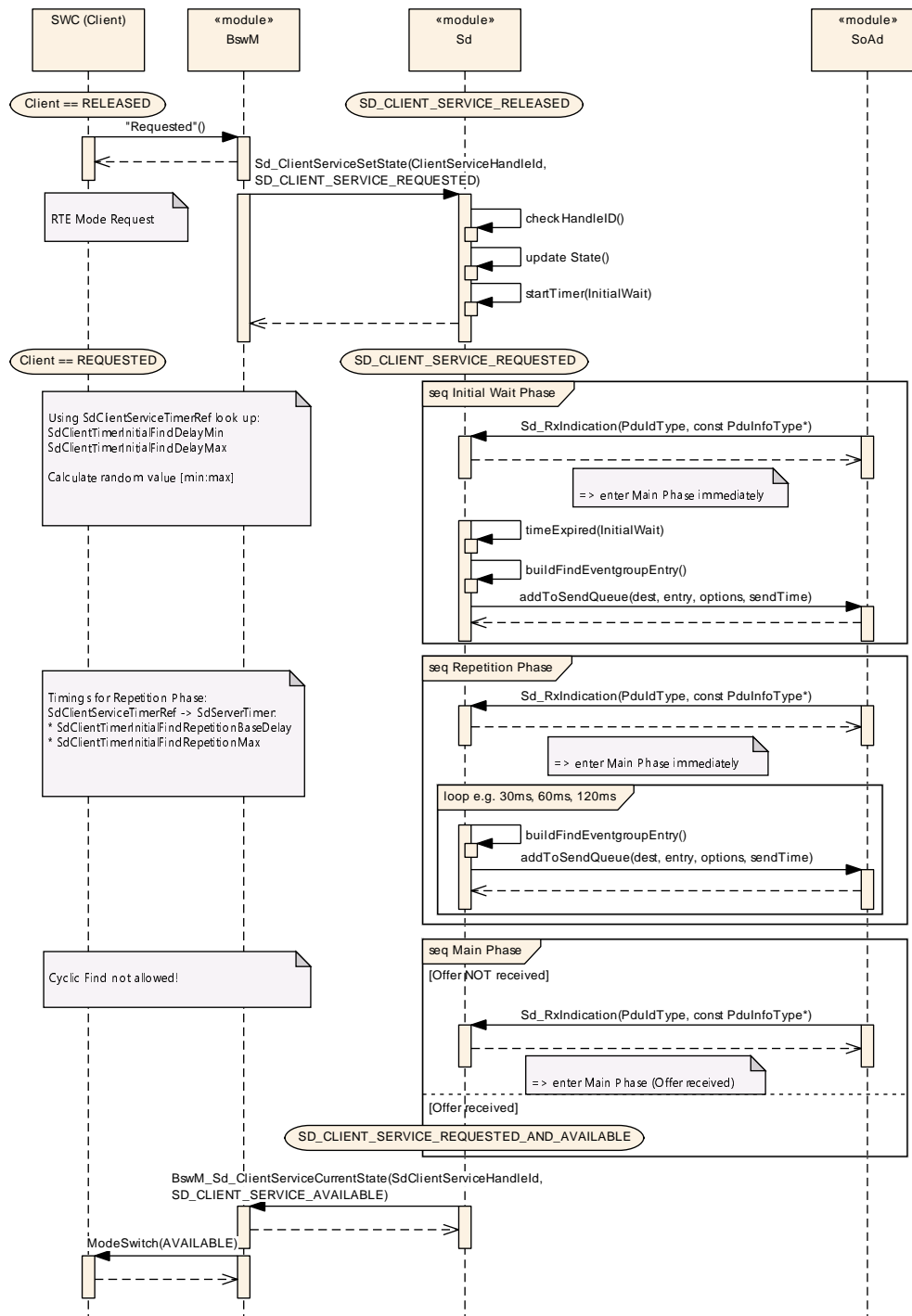


Figure 9.10: Sequence CLIENT: Start

## 9.11 ACL: Service Offer

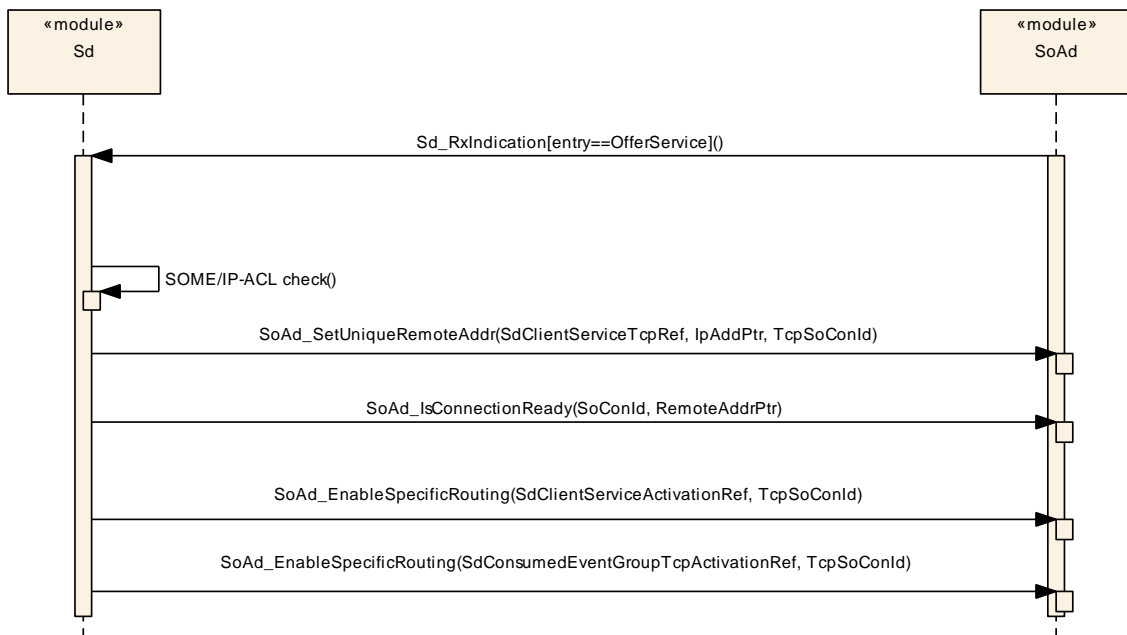


Figure 9.11: ACL check Sequence for Service Offer

## 9.12 ACL: SubscribeEventgroup

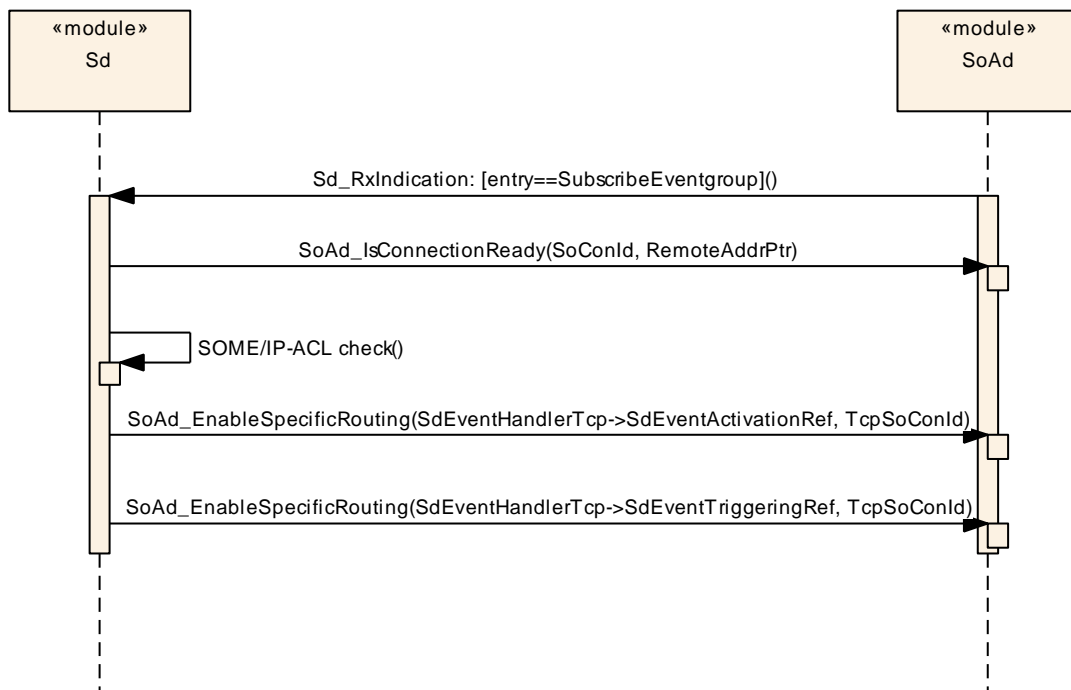
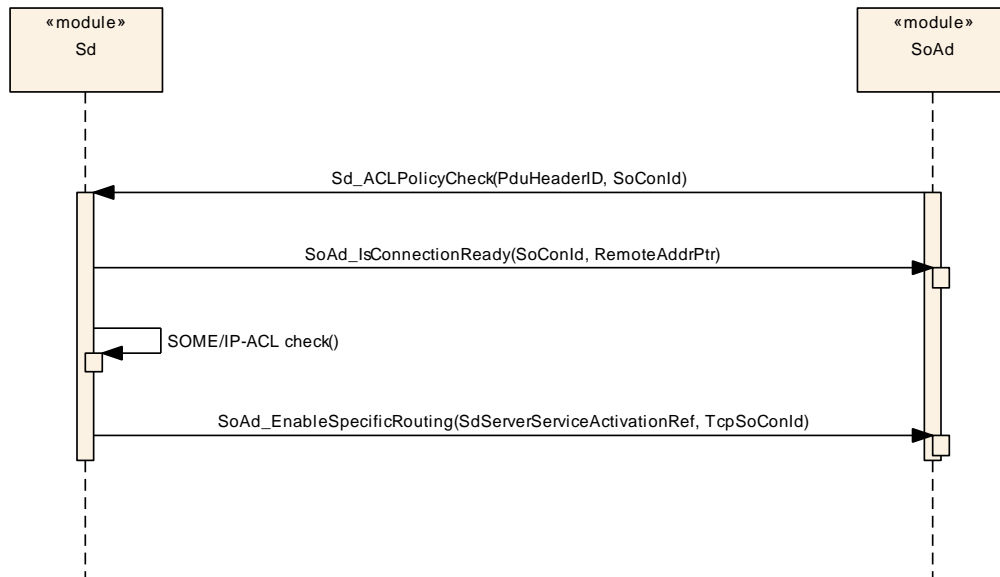


Figure 9.12: ACL check Sequence for SubscribeEventgroup

## 9.13 ACL: Method call request



**Figure 9.13: ACL check Sequence for Method Call Request**

## 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 SD.

Chapter 10.3 specifies published information of the module SD.

**[SWS\_Sd\_00135]** [The Service Discovery module shall support tool based configuration.]()

**[SWS\_Sd\_00136]** [The configuration tool shall check the consistency of the configuration parameters at system configuration time.]()

**[SWS\_Sd\_00459]** [For all SD messages sent and received via the Socket Adaptor module, the header mode shall be activated.]()

**[SWS\_Sd\_00460]** [For all SD messages sent and received via the Socket Adaptor module, the *SoAdTxPduHeaderId* and the *SoAdRxPduHeaderId* shall be set to 0xFFFF8100 respectively by Socket Adaptor.]()

**Note:** This ensures that the SoAd creates the first part of the SOME/IP header (32bit Message ID followed by a 32bit Length field) as needed for SOME/IP-SD. The remainder of the SD messages is created by this module (see chapter 7.3).

### 10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS\_BSWGeneral.

### 10.2 Containers and configuration parameters

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



## 10.2.1 Sd

<b>SWS Item</b>	<b>[ECUC_SD_00001]</b>
<b>Module Name</b>	Sd
<b>Description</b>	Configuration of the Service Discovery module.
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">SdConfig</a>	1	This container contains the configuration parameters and sub containers of the AUTOSAR Service Discovery module.
<a href="#">SdGeneral</a>	1	This container lists the general configuration parameters for the Service Discovery module.

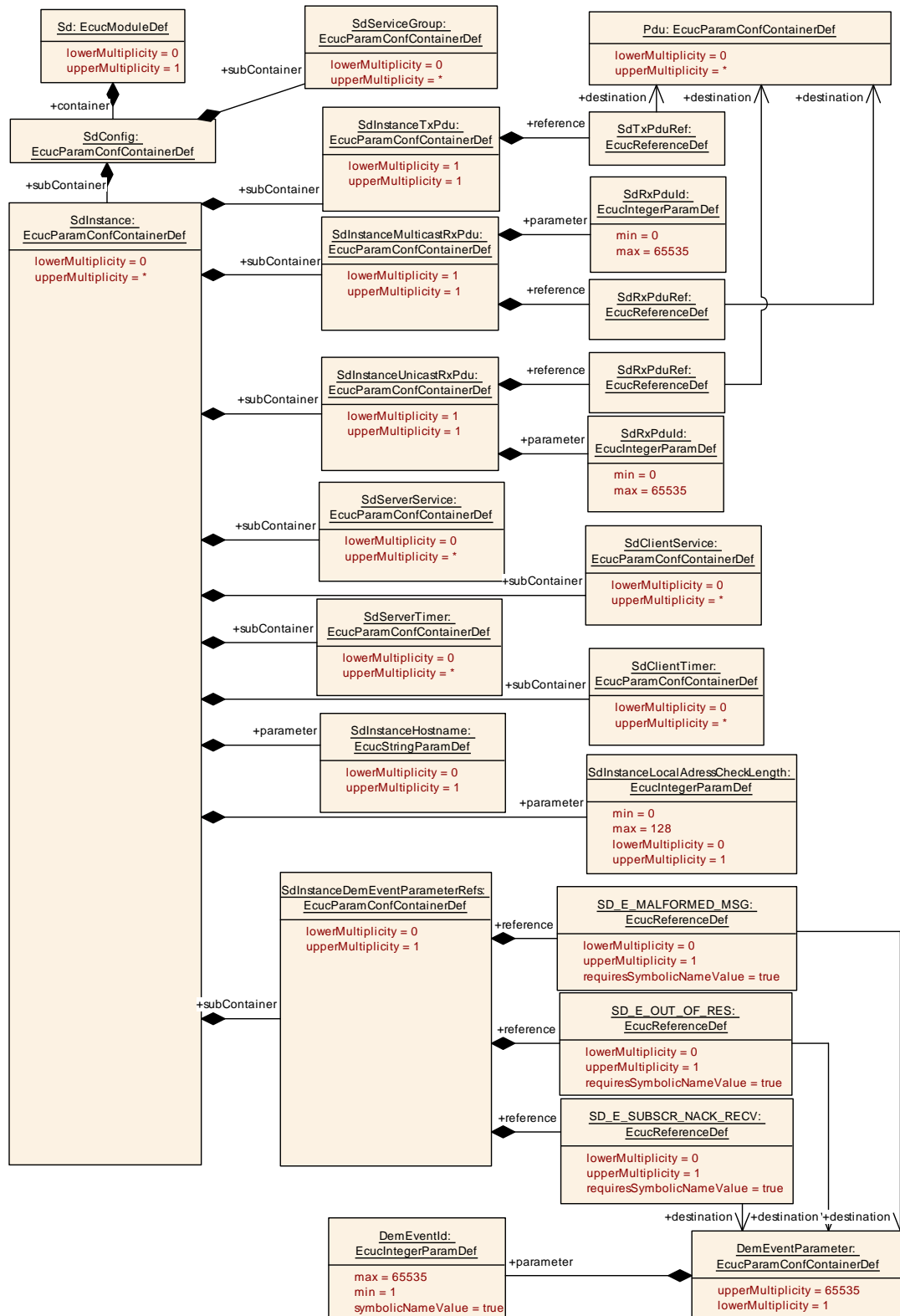


Figure 10.1: Sd Container

## 10.2.2 SdGeneral

<b>SWS Item</b>	[ECUC_SD_00002]
<b>Container Name</b>	SdGeneral
<b>Parent Container</b>	<a href="#">Sd</a>
<b>Description</b>	This container lists the general configuration parameters for the Service Discovery module.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	[ECUC_SD_00006]		
<b>Parameter Name</b>	SdDevErrorDetect		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		
<b>Description</b>	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>• true: detection and notification is enabled.</li> <li>• false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	[ECUC_Sd_00146]		
<b>Parameter Name</b>	SdEnableAclPolicyCheck		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		
<b>Description</b>	Switches the Sd AclPolicy check: <ul style="list-style-type: none"> <li>• true: feature is enabled.</li> <li>• false: feature is disabled.</li> </ul> <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU  dependency: The value of this parameter shall be set to the same value as the SoAd EnableAclCheck parameter (feature needs to be acitvated/deactivated in both modules).		

<b>SWS Item</b>	[ECUC_Sd_00157]		
<b>Parameter Name</b>	SdEnableSecurityEventReporting		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		





<b>Description</b>	Switches the reporting of security events to the IdsM: <ul style="list-style-type: none"> <li>• true: reporting is enabled.</li> <li>• false: reporting is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00008]</b>		
<b>Parameter Name</b>	SdMainFunctionCycleTime		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		
<b>Description</b>	This parameter defines the cycle time in seconds of the periodic calling of Sd main function.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	]0 .. INF[		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00139]</b>		
<b>Parameter Name</b>	SdSetRemAddrOfClientRxMulticastSoCon		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		
<b>Description</b>	<p>If SdSetRemAddrOfClientRxMulticastSoCon is set to TRUE, the Service Discovery module shall choose an multicast socket connection which match to the received Endpoint option of the corresponding OfferService. If no particular socket connection exist, then an unused socket connection with its remote address set to wildcard shall be used and the remote address shall be updated accordingly. If SdSetRemAddrOfClientRxMulticastSoCon is set to FALSE, the Service Discovery shall choose an unused socket connection with its remote address set to wildcard and skip to update the remote address, i.e. the wildcard for the remote address is kept.</p> <p>Note: setting SdSetRemAddrOfClientRxMulticastSoCon to FALSE supports the re-use of a multicast socket connection for multiple ClientServices which are located on the same ECU and subscribed to ServerServices which are located on different ECUs. The configuration of the ECU where the ClientServices are located, could be simplified by only configuring one socket connection within the multicast socket connection group.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	true		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	





<b>Scope / Dependency</b>	scope: ECU  dependency: If SdSetRemAddrOfClientRxMulticastSoCon is set to FALSE, then all affected Socket Connections shall set SoAdSocketMsgAcceptanceFilterEnabled to FALSE. Please note, a socket connection with SoAdSocketMsgAcceptanceFilter Enabled set to FALSE, accept all received events without checking the remote source address.
---------------------------	---

<b>SWS Item</b>	<b>[ECUC_SD_00131]</b>		
<b>Parameter Name</b>	SdSubscribeEventgroupRetryEnable		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		
<b>Description</b>	Switch to enable or disable the retry functionality to subscribe to Eventgroups of Server Services with TTL set to 0xFFFFF.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00007]</b>		
<b>Parameter Name</b>	SdVersionInfoApi		
<b>Parent Container</b>	<a href="#">SdGeneral</a>		
<b>Description</b>	Enables and disables the version info API.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">SdSecurityEventRefs</a>	1	Container for the references to IdsMEvent elements representing the security events that the SD module shall report to the IdsM in case the corresponding security related event occurs (and if Sd EnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events.

<b>SWS Item</b>	<b>[ECUC_Sd_00150]</b>
<b>Container Name</b>	SdSecurityEventRefs
<b>Parent Container</b>	<a href="#">SdGeneral</a>





<b>Description</b>	Container for the references to IdsMEvent elements representing the security events that the SD module shall report to the IdsM in case the corresponding security related event occurs (and if SdEnableSecurityEventReporting is set to "true"). The standardized security events in this container can be extended by vendor-specific security events.
<b>Post-Build Variant Multiplicity</b>	false
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>[ECUC_Sd_00156]</b>		
<b>Parameter Name</b>	SEV_DUPLICATE_OFFER		
<b>Parent Container</b>	<a href="#">SdSecurityEventRefs</a>		
<b>Description</b>	SD rejected Offer for a ServiceInstance which is already offered by a different endpoint and TTL still valid. <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to IdsMEvent		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_Sd_00152]</b>		
<b>Parameter Name</b>	SEV_SOME_IP_ACL_CHECK_FAILED_EVENT_SUBSCRIPTION		
<b>Parent Container</b>	<a href="#">SdSecurityEventRefs</a>		
<b>Description</b>	ACL check for a subscribe event group request failed. <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to IdsMEvent		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_Sd_00153]</b>		
<b>Parameter Name</b>	SEV_SOME_IP_ACL_CHECK_FAILED_METHOD_REQUEST		
<b>Parent Container</b>	<a href="#">SdSecurityEventRefs</a>		





<b>Description</b>	ACL check for a method request failed. <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to IdsMEvent		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_Sd_00151]</b>		
<b>Parameter Name</b>	SEV_SOME_IP_ACL_CHECK_FAILED_OFFER		
<b>Parent Container</b>	<a href="#">SdSecurityEventRefs</a>		
<b>Description</b>	ACL check for a service offer failed. <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to IdsMEvent		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

No Included Containers

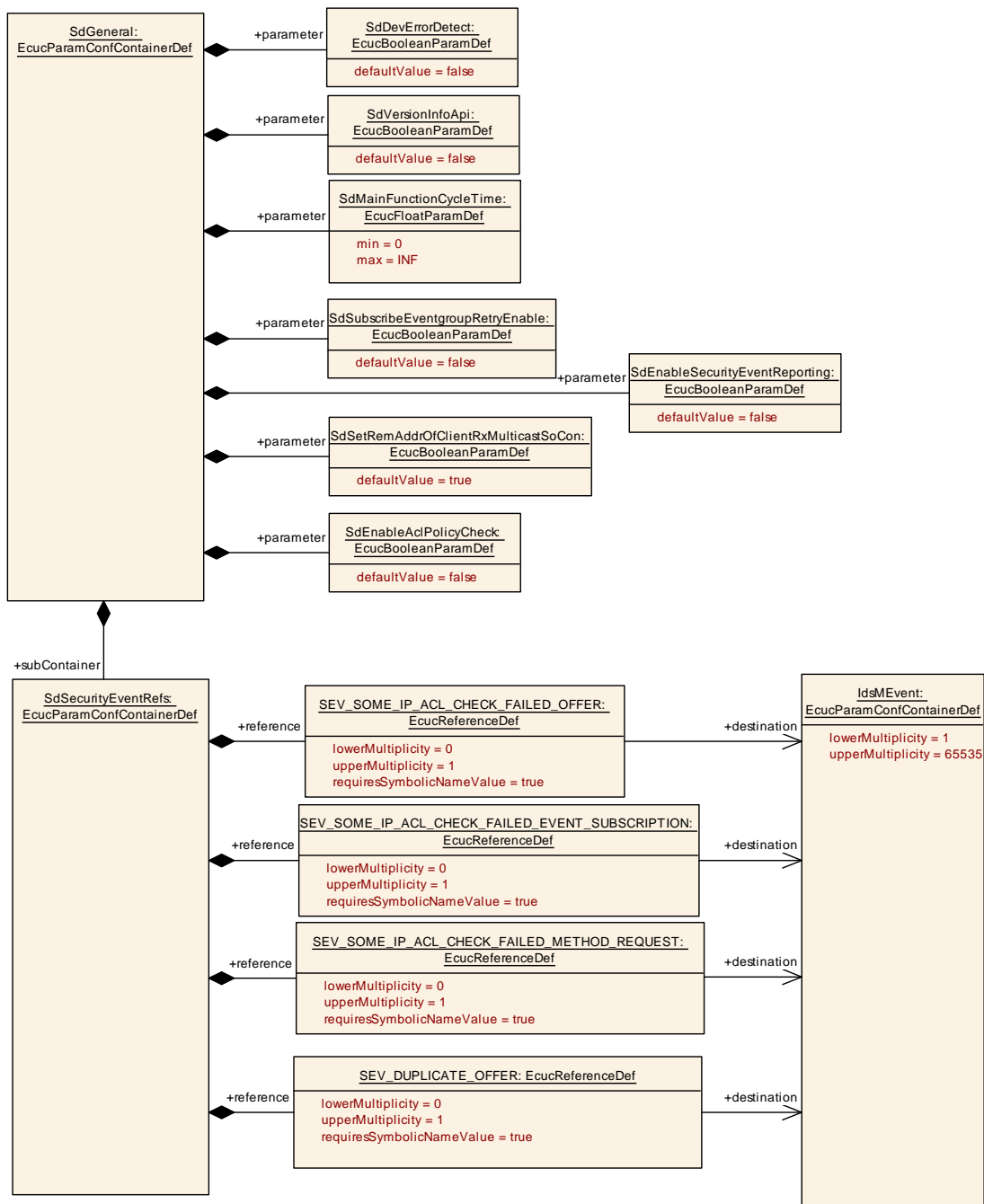


Figure 10.2: SdGeneral Container

### 10.2.3 SdConfig

SWS Item	[ECUC_SD_00003]
Container Name	SdConfig
Parent Container	<a href="#">Sd</a>







<b>Description</b>	This container contains the configuration parameters and sub containers of the AUTOSAR Service Discovery module.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>[ECUC_Sd_00154]</b>		
<b>Parameter Name</b>	SdAclCheckBlockDescriptorRef		
<b>Parent Container</b>	<a href="#">SdConfig</a>		
<b>Description</b>	Reference to the Nvm block description in the Nvm module configuration in which the Acl will be stored. <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to NvMBlockDescriptor		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
<a href="#">SdCapabilityRecordMatchCallout</a>	0..*	Callout that is invoked by the Sd implementation to determine whether the configuration options contained in the entries of a received SD message match the capability record elements configured in SdServerCapabilityRecord or SdClientCapabilityRecord.
<a href="#">SdInstance</a>	0..*	This container represents an instance of the SD; i.e. the SD configuration for a certain link.
<a href="#">SdServiceGroup</a>	0..*	This container represents a group of ClientServices and Server Services, respectively.

## 10.2.4 SdCapabilityRecordMatchCallout

<b>SWS Item</b>	<b>[ECUC_SD_00124]</b>
<b>Container Name</b>	SdCapabilityRecordMatchCallout
<b>Parent Container</b>	<a href="#">SdConfig</a>
<b>Description</b>	Callout that is invoked by the Sd implementation to determine whether the configuration options contained in the entries of a received SD message match the capability record elements configured in SdServerCapabilityRecord or SdClientCapabilityRecord.
<b>Post-Build Variant Multiplicity</b>	false
<b>Configuration Parameters</b>	

SWS Item	[ECUC_SD_00125]		
Parameter Name	SdCapabilityRecordMatchCalloutName		
Parent Container	<a href="#">SdCapabilityRecordMatchCallout</a>		
Description	Function name (i.e., C-identifier) of the SdCapabilityRecordMatchCallout.		
Multiplicity	1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

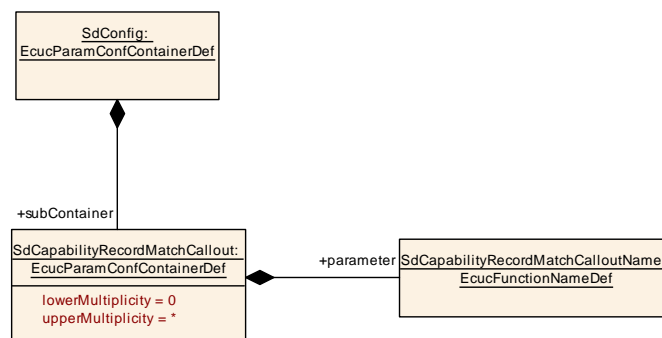


Figure 10.3: SdCapabilityRecordMatchCallout Container

## 10.2.5 SdServiceGroup

SWS Item	[ECUC_SD_00134]		
Container Name	SdServiceGroup		
Parent Container	<a href="#">SdConfig</a>		
Description	Contains the configuration parameters of the AUTOSAR SD module's SdServiceGroup S.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD





Configuration Parameters			
SWS Item	[ECUC_SD_00135]		
Parameter Name	SdServiceGroupHandleId		
Parent Container	<a href="#">SdServiceGroup</a>		
Description	The numerical value used as the ID of this SdServiceGroup. The SdServiceHandleId is required by the API calls to start and stop SdServiceGroupS.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		
No Included Containers			

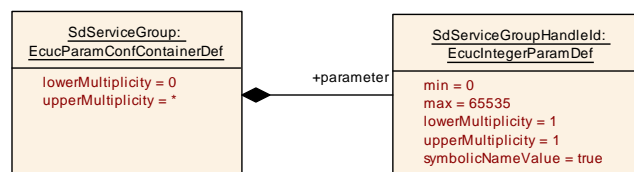


Figure 10.4: SdServiceGroup Container

## 10.2.6 SdInstance

SWS Item	[ECUC_SD_00084]
Container Name	SdInstance
Parent Container	<a href="#">SdConfig</a>
Description	This container represents an instance of the SD; i.e. the SD configuration for a certain link.
Configuration Parameters	

SWS Item	[ECUC_SD_00012]
Parameter Name	SdInstanceHostname
Parent Container	<a href="#">SdInstance</a>
Description	Configuration parameter to specify the Hostname.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Post-Build Variant Multiplicity	true





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

SWS Item	[ECUC_SD_00128]		
Parameter Name	SdInstanceLocalAdressCheckLength		
Parent Container	<a href="#">SdInstance</a>		
Description	This item describes on how many bits of the addresses shall be compared to determine, if a remote address is acceptable to be used. This shall support IPv4 (0..32) and IPv6 (0..128). If this item is not present, the security checks use the configured netmask instead. "0" meaning not to check at all. For example "8" means that the first 8 bits of a remote address must be equal to the local address to be considered acceptable.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 128		
Default value	—		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
<a href="#">SdClientService</a>	0..*	This container specifies all parameters used by Client services.
<a href="#">SdClientTimer</a>	0..*	This container specifies all timers used by the Service Discovery module for Client Services.
<a href="#">SdInstanceDemEventParameter Refs</a>	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
<a href="#">SdInstanceMulticastRxPdu</a>	1	This container specifies the received PDU.
<a href="#">SdInstanceTxPdu</a>	1	This container specifies the transmitted PDU.
<a href="#">SdInstanceUnicastRxPdu</a>	1	This container specifies the received PDU.
<a href="#">SdServerService</a>	0..*	This container specifies all parameters used by Server services.
<a href="#">SdServerTimer</a>	0..*	This container specifies all timers used by the Service Discovery module for Server Services.

## 10.2.7 SdClientService

SWS Item	[ECUC_SD_00005]
Container Name	SdClientService
Parent Container	<a href="#">SdInstance</a>
Description	This container specifies all parameters used by Client services.
Configuration Parameters	

SWS Item	[ECUC_SD_00143]		
Parameter Name	SdClientServiceAutoRequire		
Parent Container	<a href="#">SdClientService</a>		
Description	If existing and set to true, this Service will be set to "required" on start.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local  dependency: SdClientServiceAutoRequire could only be set to true, if the SdClient Service is NOT referencing a SdServiceGroup		

SWS Item	[ECUC_SD_00079]		
Parameter Name	SdClientServiceHandleId		
Parent Container	<a href="#">SdClientService</a>		
Description	The HandleId by which the BswM can identify this Client Service Instance.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local  withAuto = true		

SWS Item	[ECUC_SD_00020]		
Parameter Name	SdClientServiceId		
Parent Container	<a href="#">SdClientService</a>		
Description	Id to identify the service. This is unique for the service interface.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65534		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE





	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00022]</b>		
<b>Parameter Name</b>	SdClientServiceInstanceId		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	Configuration parameter to specify Instance Id of the service as used in SD entries.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65534		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00070]</b>		
<b>Parameter Name</b>	SdClientServiceMajorVersion		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	Major version number of the Service as used in the SD entries.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 254		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00071]</b>		
<b>Parameter Name</b>	SdClientServiceMinorVersion		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	Minor version number of the Service as used in the SD Service Entries. If configured to 0xffffffff (any), SD will accept all Minor Versions.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967295		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

SWS Item	[ECUC_SD_00140]		
Parameter Name	SdVersionDrivenFindBehavior		
Parent Container	<a href="#">SdClientService</a>		
Description	Defined the possible acceptance kinds for required service instances. <b>Tags:</b> atp.Status=draft		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	EXACT_OR_ANY_MINOR_VERSION	Search for ANY or specific minor version service instance and select either ALL returned service instances (in case of ANY) or exactly the specific minor version service instances defined in Sd ClientServiceMinorVersion.	
	MINIMUM_MINOR_VERSION	Search for ANY minor version service instance and select only those service instances which have an equal or greater minor version than given in SdClientServiceMinorVersion.	
Default value	<a href="#">EXACT_OR_ANY_MINOR_VERSION</a>		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00127]		
Parameter Name	SdClientCapabilityRecordMatchCalloutRef		
Parent Container	<a href="#">SdClientService</a>		
Description	Reference to a SdCapabilityRecordMatchCallout, The referenced SdCapabilityRecord MatchCallout is invoked to determine whether the configuration options contained in the entries of a received SD message match the client's configured SdClientCapability Record elements.		
Multiplicity	0..1		
Type	Reference to <a href="#">SdCapabilityRecordMatchCallout</a>		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_Sd_00145]		
Parameter Name	SdClientServiceMulticastRef		
Parent Container	<a href="#">SdClientService</a>		





<b>Description</b>	<p>Reference to the SoAdSocketConnection representing the data path (UDP) for communication with the server. This element is also used to set the remote address of the server.</p> <p>This is used, if a ClientService subscribes with a Consumed Eventgroup multicast endpoint. This is an alternative to subscribe with a Consumed Eventgroup unicast endpoint (see SdClientServiceUdpRef).</p> <p>Please note: usage of this reference is mutually exclusive to SdClientServiceUdpRef.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to SoAdSocketConnectionGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	<p>scope: local</p> <p>dependency: This parameter is only valid if SdClientServiceUdpRef is NOT configured.</p>		

<b>SWS Item</b>	<b>[ECUC_SD_00100]</b>		
<b>Parameter Name</b>	SdClientServiceTcpRef		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	<p>Reference to the SoAdSocketConnection representing the data path (TCP) for communication with methods.</p> <p>This element is also used to set the remote address of the server and to open the TCP connection.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to SoAdSocketConnectionGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00103]</b>		
<b>Parameter Name</b>	SdClientServiceTimerRef		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	The reference of the SdClientTimer container for this service.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to <a href="#">SdClientTimer</a>		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME







	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00101]</b>		
<b>Parameter Name</b>	SdClientServiceUdpRef		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	<p>Reference to the SoAdSocketConnection representing the data path (UDP) for communication with methods.</p> <p>This element is also used to set the remote address of the server.</p> <p>This is used, if a ClientService subscribes with a Consumed Eventgroup unicast endpoint. This is an alternative to subscribe with a Consumed Eventgroup multicast endpoint. (see SdClientServiceMulticastRef).</p> <p>Please note: usage of this reference is mutually exclusive to SdClientServiceMulticastRef.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to SoAdSocketConnectionGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local  dependency: This parameter is only valid if SdClientServiceMulticastRef is NOT configured.		

<b>SWS Item</b>	<b>[ECUC_SD_00137]</b>		
<b>Parameter Name</b>	SdServiceGroupRef		
<b>Parent Container</b>	<a href="#">SdClientService</a>		
<b>Description</b>	Reference to the SdServiceGroupS this SdClientService belongs to.		
<b>Multiplicity</b>	0..*		
<b>Type</b>	Reference to <a href="#">SdServiceGroup</a>		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">SdBlocklistedVersions</a>	0..1	Collection of blocklisted versions. <b>Tags:</b> atp.Status=draft
<a href="#">SdClientCapabilityRecord</a>	0..*	Sd uses capability records to store arbitrary name/value pairs conveying additional information about the named service. The following use cases are supported: 1) Key present, with no value (e.g. "passreq" – password required for this service) 2) Key present, with empty value (e.g. "PlugIns=" server supports plugins, but none are presently installed) 3) Key present, with non-empty value (e.g. "PlugIns=JPEG,MPEG2,MPEG4")
<a href="#">SdClientServiceAllowedProvider</a>	0..1	The container defines the allowed provider for this ClientService. <b>Tags:</b> atp.Status=draft
<a href="#">SdConsumedEventGroup</a>	0..*	This container specifies all parameters for consumed event groups.
<a href="#">SdConsumedMethods</a>	0..1	Container element for representing the data path for accessing the server methods.

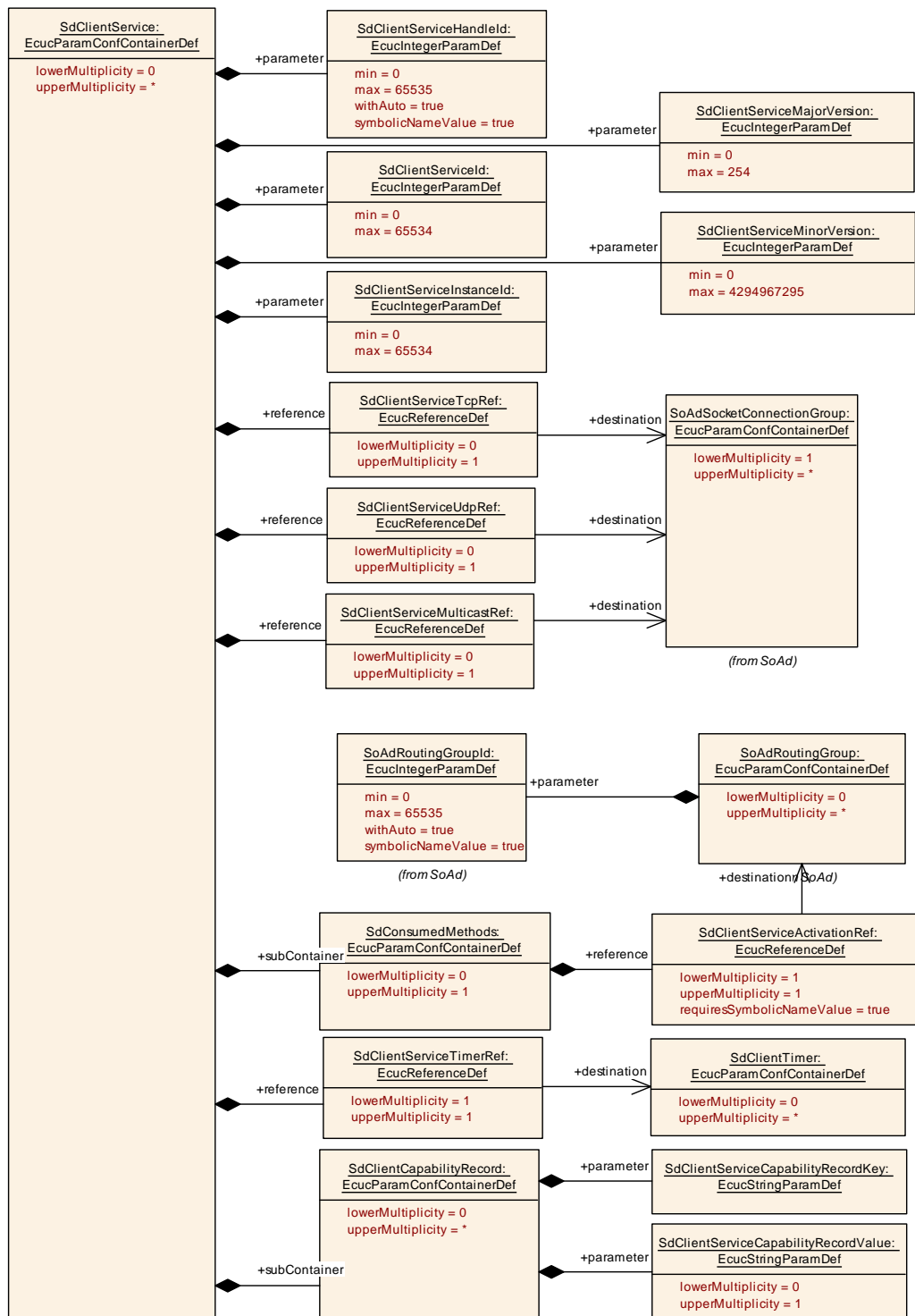


Figure 10.5: SdClientService Container

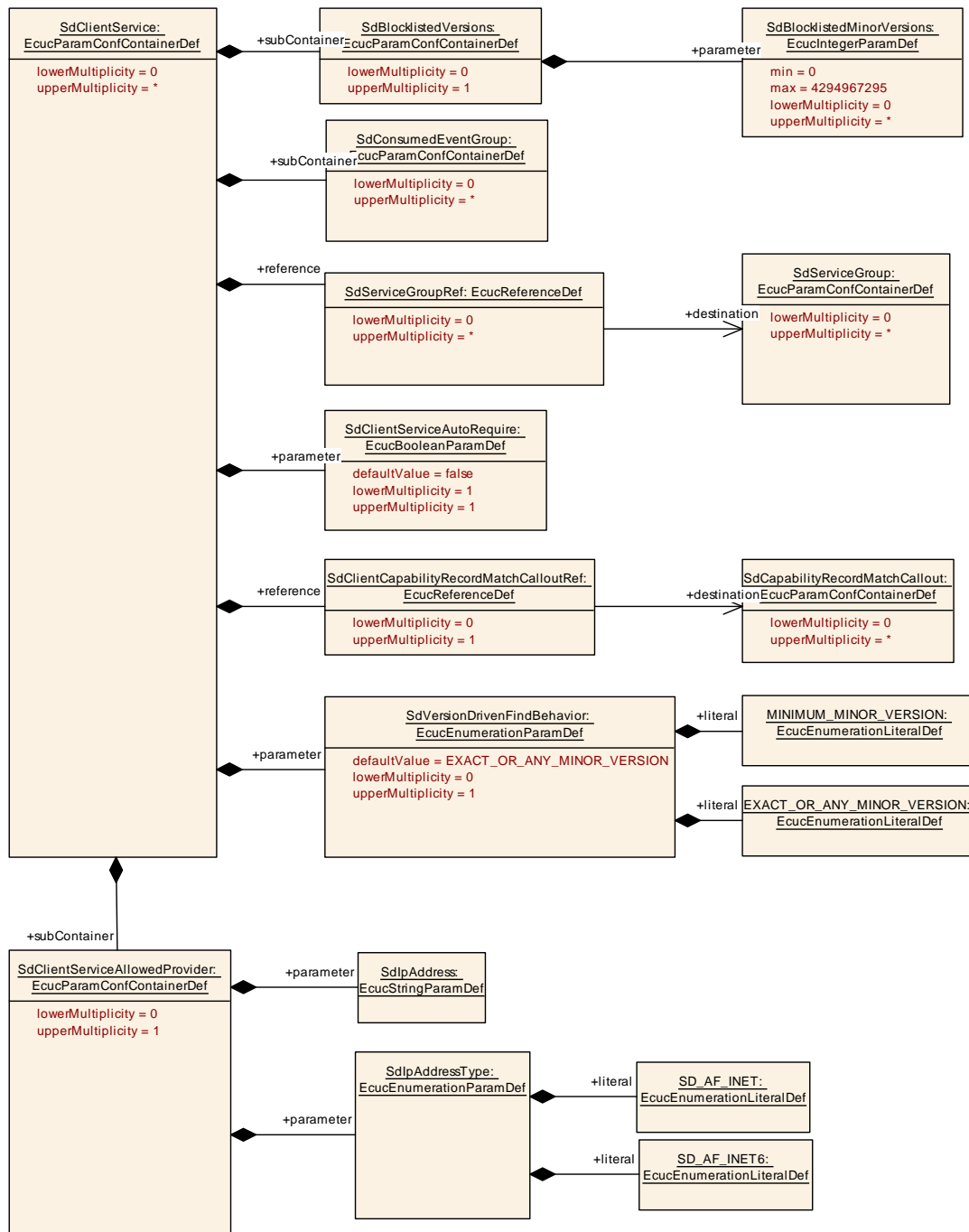


Figure 10.6: SdClientService Container

## 10.2.8 SdBlocklistedVersions

SWS Item	[ECUC_SD_00141]		
Container Name	SdBlocklistedVersions		
Parent Container	<a href="#">SdClientService</a>		
Description	Collection of blocklisted versions. <b>Tags:</b> atp.Status=draft		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	[ECUC_SD_00142]		
Parameter Name	SdBlocklistedMinorVersions		
Parent Container	<a href="#">SdBlocklistedVersions</a>		
Description	Blocklisted MinorVersions. <b>Tags:</b> atp.Status=draft		
Multiplicity	0..*		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

## 10.2.9 SdClientCapabilityRecord

SWS Item	[ECUC_SD_00072]
Container Name	SdClientCapabilityRecord
Parent Container	<a href="#">SdClientService</a>





<b>Description</b>	<p>Sd uses capability records to store arbitrary name/value pairs conveying additional information about the named service.</p> <p>The following use cases are supported: 1) Key present, with no value (e.g. "passreq" – password required for this service)</p> <p>2) Key present, with empty value (e.g. "PlugIns=" server supports plugins, but none are presently installed)</p> <p>3) Key present, with non-empty value (e.g. "PlugIns=JPEG,MPEG2,MPEG4")</p>
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>[ECUC_SD_00073]</b>		
<b>Parameter Name</b>	SdClientServiceCapabilityRecordKey		
<b>Parent Container</b>	<a href="#">SdClientCapabilityRecord</a>		
<b>Description</b>	Defines a CapabilityRecord key.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucStringParamDef		
<b>Default value</b>	–		
<b>Regular Expression</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00074]</b>		
<b>Parameter Name</b>	SdClientServiceCapabilityRecordValue		
<b>Parent Container</b>	<a href="#">SdClientCapabilityRecord</a>		
<b>Description</b>	Defines the corresponding CapabilityRecord value.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucStringParamDef		
<b>Default value</b>	–		
<b>Regular Expression</b>	–		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local		

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

## 10.2.10 SdConsumedEventGroup

<b>SWS Item</b>	[ECUC_SD_00056]
<b>Container Name</b>	SdConsumedEventGroup
<b>Parent Container</b>	<a href="#">SdClientService</a>
<b>Description</b>	A Service may have event groups which can be consumed. A service consumer has to subscribe to the corresponding event-group. After the subscription the event consumer takes the role of a server and the event provider that of a client.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	[ECUC_SD_00144]		
<b>Parameter Name</b>	SdConsumedEventGroupAutoRequire		
<b>Parent Container</b>	<a href="#">SdConsumedEventGroup</a>		
<b>Description</b>	If existing and set to true, this EventGroup will be set to "required" on start.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	[ECUC_SD_00116]		
<b>Parameter Name</b>	SdConsumedEventGroupHandleId		
<b>Parent Container</b>	<a href="#">SdConsumedEventGroup</a>		
<b>Description</b>	The HandleId by which the BswM can identify this EventGroup.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: local withAuto = true		

<b>SWS Item</b>	[ECUC_SD_00057]		
<b>Parameter Name</b>	SdConsumedEventGroupId		
<b>Parent Container</b>	<a href="#">SdConsumedEventGroup</a>		
<b>Description</b>	The Eventgroup Id of this eventGroup as a unique identifier of the eventgroup in this service. This identifier is used for EventGroup entries as well. Please note, that the Eventgroup ID 0x0000 is reserved.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65534		
<b>Default value</b>	–		





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

SWS Item	[ECUC_SD_00106]		
Parameter Name	SdConsumedEventGroupMulticastActivationRef		
Parent Container	<a href="#">SdConsumedEventGroup</a>		
Description	<p>The reference of a Routing Group in order to activate and setup the Socket Connection for Multicast Events of this EventGroup. The Multicast address from the received Multicast Option is setup by SoAd_RequestIpAddrAssignment.</p> <p>The local address is the same as for the unicast events; thus, it was sent in the UDP Endpoint option of the Subscribe EventGroup entry.</p> <p>This is usually equal to the SdConsumedEventGroupUdpActivationRef.</p>		
Multiplicity	0..1		
Type	Symbolic name reference to SoAdRoutingGroup		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00119]		
Parameter Name	SdConsumedEventGroupMulticastGroupRef		
Parent Container	<a href="#">SdConsumedEventGroup</a>		
Description	Reference to the SoAdSocketConnectionGroup representing the multicast data path (UDP).		
Multiplicity	0..*		
Type	Reference to SoAdSocketConnectionGroup		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00105]		
Parameter Name	SdConsumedEventGroupTcpActivationRef		
Parent Container	<a href="#">SdConsumedEventGroup</a>		







<b>Description</b>	<p>The reference of the Routing Group for activation of the data path for receiving TCP events.</p> <p>This element is also being used for getting the IP address and port number for building the TCP endpoint option for the Subscribe EventGroup entry.</p> <p>If no TCP methods are used in the service, this element is also being used for setting the remote address (TCP Endpoint option referenced by the Offer Service entry) and opening the TCP connection to the server before sending the Subscribe EventGroup entry. If multiple EventGroups of the same Service Instance are subscribed the TCP connection will be shared and must be opened only once.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00107]</b>		
<b>Parameter Name</b>	SdConsumedEventGroupTimerRef		
<b>Parent Container</b>	<a href="#">SdConsumedEventGroup</a>		
<b>Description</b>	The reference of the SdClientTimer container for this eventGroup.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to <a href="#">SdClientTimer</a>		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00104]</b>		
<b>Parameter Name</b>	SdConsumedEventGroupUdpActivationRef		
<b>Parent Container</b>	<a href="#">SdConsumedEventGroup</a>		
<b>Description</b>	<p>The reference of the Routing Group for activation of the data path for receiving UDP events.</p> <p>This element is also being used for getting the IP address and port number for building the UDP Endpoint option or Consumed Multicast option for the Subscribe EventGroup entry.</p> <p>If no UDP methods are used in the service, this element is also being used for setting the remote address (UDP Endpoint option referenced by the Offer Service entry). If multiple EventGroups of the same Service Instance are subscribed the UDP Socket Connection will be shared and must be set only once.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		





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

#### No Included Containers

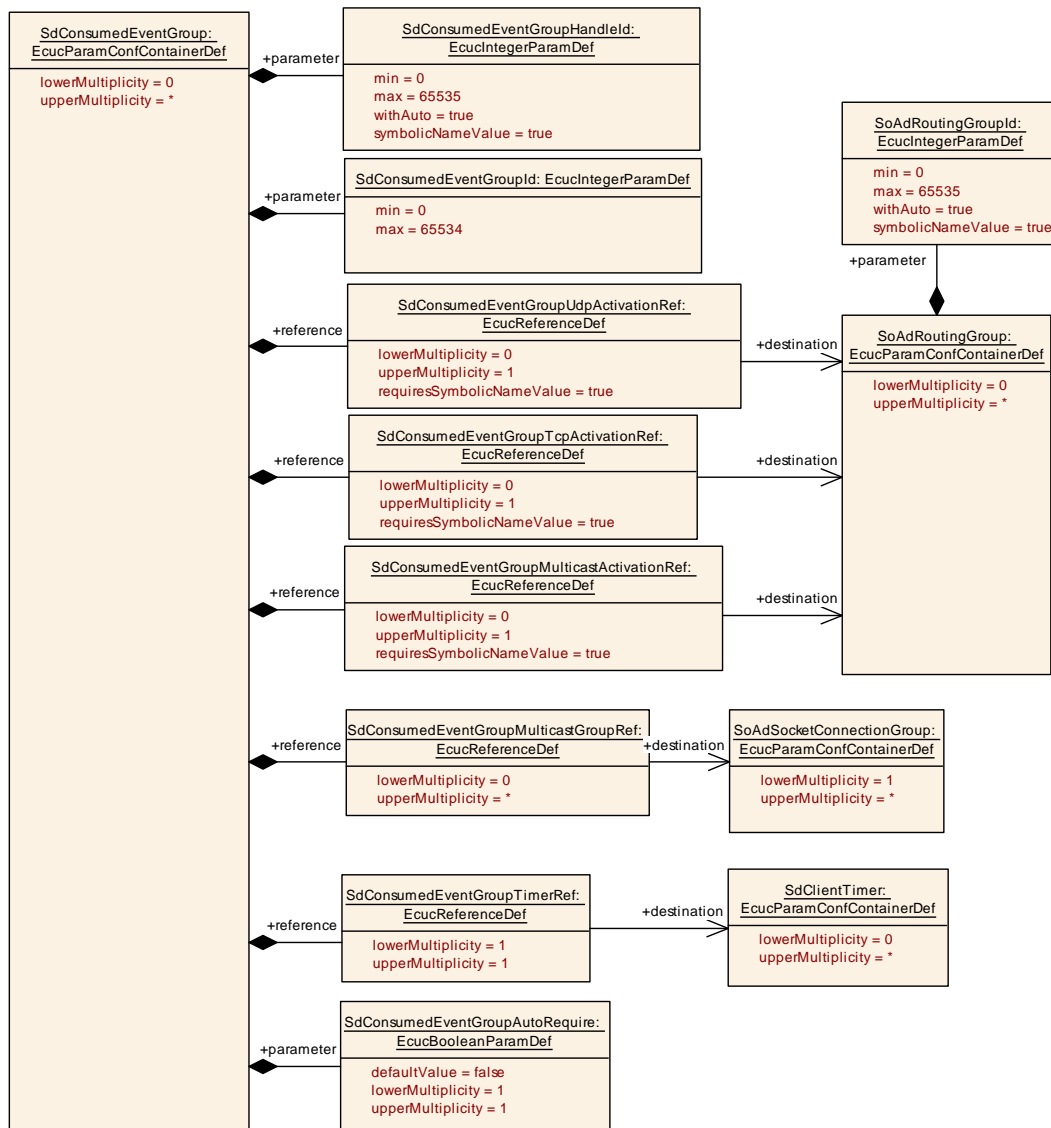


Figure 10.7: SdConsumedEventGroup Container

### 10.2.11 SdConsumedMethods

SWS Item	[ECUC_SD_00099]		
Container Name	SdConsumedMethods		
Parent Container	<a href="#">SdClientService</a>		
Description	Container element for representing the data path for accessing the server methods.		
Configuration Parameters			

SWS Item	[ECUC_SD_00102]		
Parameter Name	SdClientServiceActivationRef		
Parent Container	<a href="#">SdConsumedMethods</a>		
Description	Reference to a SoAdRoutingGroupRef to activate/deactivate the data path for the methods.		
Multiplicity	1		
Type	Symbolic name reference to SoAdRoutingGroup		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers
------------------------

### 10.2.12 SdClientTimer

SWS Item	[ECUC_SD_00043]		
Container Name	SdClientTimer		
Parent Container	<a href="#">SdInstance</a>		
Description	This container specifies all timers used by the Service Discovery module for Client Services.		
Configuration Parameters			

SWS Item	[ECUC_SD_00063]		
Parameter Name	SdClientTimerInitialFindDelayMax		
Parent Container	<a href="#">SdClientTimer</a>		
Description	Max value in [s] to delay randomly the transmission of a find message. This parameter is mandatory for ClientService.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD





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

SWS Item	[ECUC_SD_00044]		
Parameter Name	SdClientTimerInitialFindDelayMin		
Parent Container	<a href="#">SdClientTimer</a>		
Description	Min value in [s] to delay randomly the transmission of a find message. This parameter is mandatory for ClientService.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_SD_00047]		
Parameter Name	SdClientTimerInitialFindRepetitionsBaseDelay		
Parent Container	<a href="#">SdClientTimer</a>		
Description	The base delay in [s] for find repetitions. Successive finds have an exponential back off delay (1x base delay, 2x base delay, 4x base delay, ...). This parameter is mandatory for ClientService.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_SD_00046]		
Parameter Name	SdClientTimerInitialFindRepetitionsMax		
Parent Container	<a href="#">SdClientTimer</a>		





<b>Description</b>	Configuration for the maximum number of find repetitions. This parameter is mandatory for ClientService.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 10		
<b>Default value</b>	–		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00036]</b>		
<b>Parameter Name</b>	SdClientTimerRequestResponseMaxDelay		
<b>Parent Container</b>	<a href="#">SdClientTimer</a>		
<b>Description</b>	Maximum allowable response delay to entries received by multicast in seconds. This parameter is mandatory for ConsumedEventGroups.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	–		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00064]</b>		
<b>Parameter Name</b>	SdClientTimerRequestResponseMinDelay		
<b>Parent Container</b>	<a href="#">SdClientTimer</a>		
<b>Description</b>	Minimum allowable response delay to the find message in seconds. This parameter is mandatory for ConsumedEventGroups.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	–		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME





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

<b>SWS Item</b>	<b>[ECUC_SD_00075]</b>		
<b>Parameter Name</b>	SdClientTimerTTL		
<b>Parent Container</b>	<a href="#">SdClientTimer</a>		
<b>Description</b>	Time to live for find and subscribe messages. Note! The TTL value for find messages shall be ignored by the server service and the configuration is only kept for backward compatibility		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 16777215		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00133]</b>		
<b>Parameter Name</b>	SdSubscribeEventgroupRetryDelay		
<b>Parent Container</b>	<a href="#">SdClientTimer</a>		
<b>Description</b>	Time in seconds when a subscription to an event group shall be retriggered, if no SubscribeEventGroupAck or SubscribeEventGroupNack was received.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0.001 .. 50]		
<b>Default value</b>	0.01		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local  dependency: SdSubscribeEventgroupRetryDelay is only applicable if SdSubscribeEventgroupRetryEnable is set to TRUE and SdSubscribeEventgroupRetryMax > 0.		

<b>SWS Item</b>	<b>[ECUC_SD_00132]</b>		
<b>Parameter Name</b>	SdSubscribeEventgroupRetryMax		
<b>Parent Container</b>	<a href="#">SdClientTimer</a>		





<b>Description</b>	Maximum count of retry a subscription, if a subscription to an event group is not acknowledged by SubscribeEventGroupAck or SubscribeEventGroupNack. 0x0=no retry, 0xFF=retry forever (as long as the event group is requested)		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	0		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local  dependency: SdSubscribeEventgroupRetryMax is only applicable if SdSubscribeEventgroupRetryEnable is set to TRUE		

No Included Containers

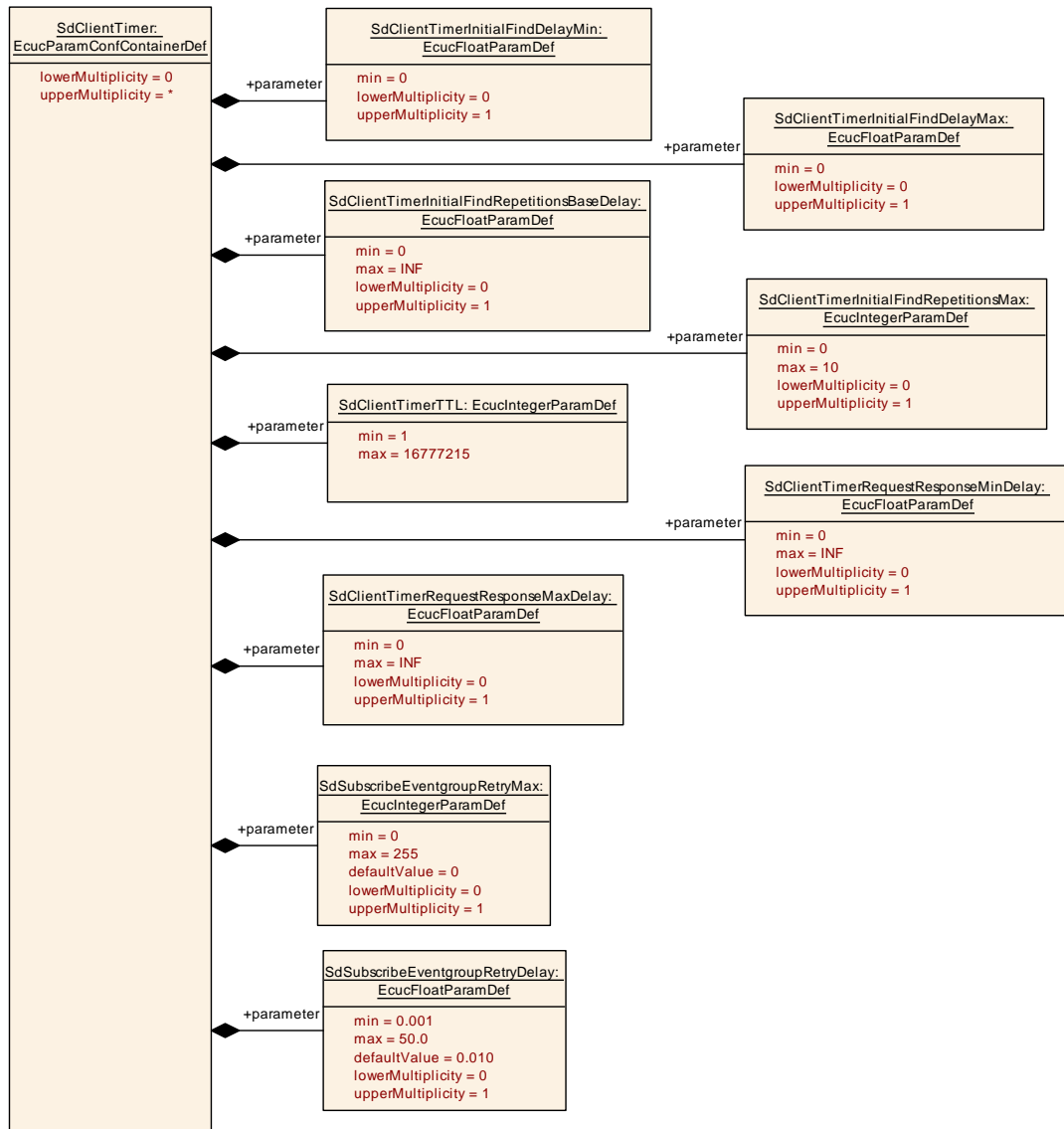


Figure 10.8: SdClientTimer Container

### 10.2.13 SdInstanceDemEventParameterRefs

SWS Item	[ECUC_SD_00120]
Container Name	SdInstanceDemEventParameterRefs
Parent Container	<a href="#">SdInstance</a>
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Configuration Parameters	



SWS Item	[ECUC_SD_00121]		
Parameter Name	SD_E_MALFORMED_MSG		
Parent Container	<a href="#">SdInstanceDemEventParameterRefs</a>		
Description	Reference to the DemEventParameter which shall be issued when the SD Instance received malformed message.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00122]		
Parameter Name	SD_E_OUT_OF_RES		
Parent Container	<a href="#">SdInstanceDemEventParameterRefs</a>		
Description	Reference to the DemEventParameter which shall be issued when the SD Instance does not have enough resources to handle client.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00123]		
Parameter Name	SD_E_SUBSCR_NACK_RECV		
Parent Container	<a href="#">SdInstanceDemEventParameterRefs</a>		
Description	Reference to the DemEventParameter which shall be issued when receiving Subscribe EventgroupNack entry.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME





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

No Included Containers

### 10.2.14 SdInstanceMulticastRxPdu

<b>SWS Item</b>	<b>[ECUC_SD_00081]</b>		
<b>Container Name</b>	SdInstanceMulticastRxPdu		
<b>Parent Container</b>	<a href="#">SdInstance</a>		
<b>Description</b>	This container specifies the received PDU.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>[ECUC_SD_00028]</b>		
<b>Parameter Name</b>	SdRxPduId		
<b>Parent Container</b>	<a href="#">SdInstanceMulticastRxPdu</a>		
<b>Description</b>	ID of the PDU that will be received via the API Sd_SoAdIfRxIndication().		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00029]</b>		
<b>Parameter Name</b>	SdRxPduRef		
<b>Parent Container</b>	<a href="#">SdInstanceMulticastRxPdu</a>		
<b>Description</b>	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to Pdu		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

No Included Containers

### 10.2.15 SdInstanceTxPdu

SWS Item	[ECUC_SD_00030]
Container Name	SdInstanceTxPdu
Parent Container	<a href="#">SdInstance</a>
Description	This container specifies the transmitted PDU.
Configuration Parameters	

SWS Item	[ECUC_SD_00109]		
Parameter Name	SdTxPduRef		
Parent Container	<a href="#">SdInstanceTxPdu</a>		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

### 10.2.16 SdInstanceUnicastRxPdu

SWS Item	[ECUC_SD_00027]
Container Name	SdInstanceUnicastRxPdu
Parent Container	<a href="#">SdInstance</a>
Description	This container specifies the received PDU.
Configuration Parameters	

SWS Item	[ECUC_SD_00082]		
Parameter Name	SdRxPduId		
Parent Container	<a href="#">SdInstanceUnicastRxPdu</a>		
Description	ID of the PDU that will be received via the API <code>Sd_SoAdIfRxIndication()</code> .		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_SD_00083]		
Parameter Name	SdRxPduRef		
Parent Container	<a href="#">SdInstanceUnicastRxPdu</a>		
Description	Reference to the "global" Pdu structure to allow harmonization of handle IDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

### 10.2.17 SdServerService

SWS Item	[ECUC_SD_00004]
Container Name	SdServerService
Parent Container	<a href="#">SdInstance</a>
Description	This container specifies all parameters used by Server services.
Configuration Parameters	

SWS Item	[ECUC_SD_00138]		
Parameter Name	SdServerServiceAutoAvailable		
Parent Container	<a href="#">SdServerService</a>		
Description	If existing and set to true, this Service will be set to "Available" on start.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		
	dependency: SdServerServiceAutoAvailable could only be set to true, if the SdServer Service is NOT referencing a SdServiceGroup		

SWS Item	[ECUC_SD_00110]		
Parameter Name	SdServerServiceHandleId		
Parent Container	<a href="#">SdServerService</a>		
Description	The HandleId by which the BswM can identify this Server Service Instance.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		





Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local withAuto = true		

SWS Item	[ECUC_SD_00009]		
Parameter Name	SdServerServiceId		
Parent Container	<a href="#">SdServerService</a>		
Description	Id to identify the service. This is unique for the service interface.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65534		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00011]		
Parameter Name	SdServerServiceInstanceId		
Parent Container	<a href="#">SdServerService</a>		
Description	Configuration parameter to specify Instance Id of the Service implemented by the Server Service.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65534		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00129]		
Parameter Name	SdServerServiceLoadBalancingPriority		
Parent Container	<a href="#">SdServerService</a>		
Description	Defines the value to be used for load balancing priority in the service offer. Lower value means higher priority.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE





	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00130]</b>		
<b>Parameter Name</b>	SdServerServiceLoadBalancingWeight		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Defines the value to be used for load balancing weight in the service offer. Higher value means higher probability to be chosen.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00068]</b>		
<b>Parameter Name</b>	SdServerServiceMajorVersion		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Major version number of the Service as used in SD Entries.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 254		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00069]</b>		
<b>Parameter Name</b>	SdServerServiceMinorVersion		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Minor version number of the Service as used e.g. in Offer Service entries.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 4294967294		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00126]</b>		
<b>Parameter Name</b>	SdServerCapabilityRecordMatchCalloutRef		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Reference to a SdCapabilityRecordMatchCallout, The referenced SdCapabilityRecordMatchCallout is invoked to determine whether the configuration options contained in the entries of a received SD message match the server's configured SdServerCapabilityRecord elements.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to <a href="#">SdCapabilityRecordMatchCallout</a>		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00088]</b>		
<b>Parameter Name</b>	SdServerServiceTcpRef		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Reference to SoAdSocketConnectionGroup used for methods.  This is used to access the local IP address and port for building the endpoint option for offers of this service.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to SoAdSocketConnectionGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00086]</b>		
<b>Parameter Name</b>	SdServerServiceTimerRef		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	The reference of the SdServerTimer container for this service.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to <a href="#">SdServerTimer</a>		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00089]</b>		
<b>Parameter Name</b>	SdServerServiceUdpRef		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Reference to SoAdSocketConnectionGroup used for methods. This is used to access the local IP address and port for building the endpoint option for offers of this service.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to SoAdSocketConnectionGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00136]</b>		
<b>Parameter Name</b>	SdServiceGroupRef		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	Reference to the SdServiceGroupS this SdServerService belongs to.		
<b>Multiplicity</b>	0..*		
<b>Type</b>	Reference to <a href="#">SdServiceGroup</a>		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">SdEventHandler</a>	0..*	Container Element for representing an EventGroup as part of the Service Instance.
<a href="#">SdProvidedMethods</a>	0..1	Container element for representing the needed elements of the data path for the methods provided by the service.
<a href="#">SdServerCapabilityRecord</a>	0..*	Sd uses capability records to store arbitrary name/value pairs conveying additional information about the named service.  The following use cases are supported: 1) Key present, with no value (e.g. "passreq" – password required for this service) 2) Key present, with empty value (e.g. "Plugins=" server supports plugins, but none are presently installed) 3) Key present, with non-empty value (e.g. "Plug Ins=JPEG,MPEG2,MPEG4")







Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">SdServerServiceAllowedConsumers</a>	0..*	This container defines a list of consumers that are allowed to access this SdServerService. <b>Tags:</b> atp.Status=draft

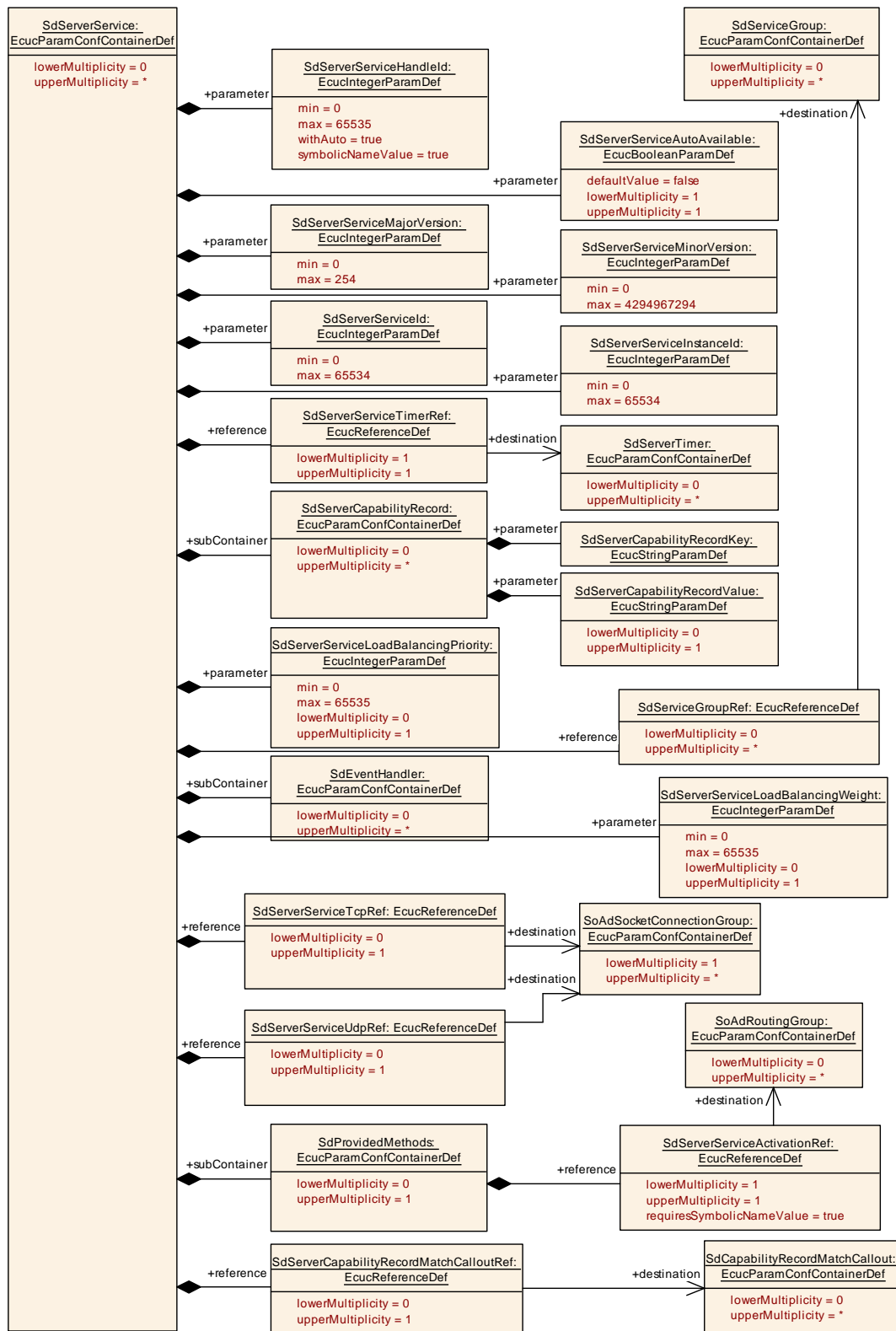


Figure 10.9: SdServerService Container

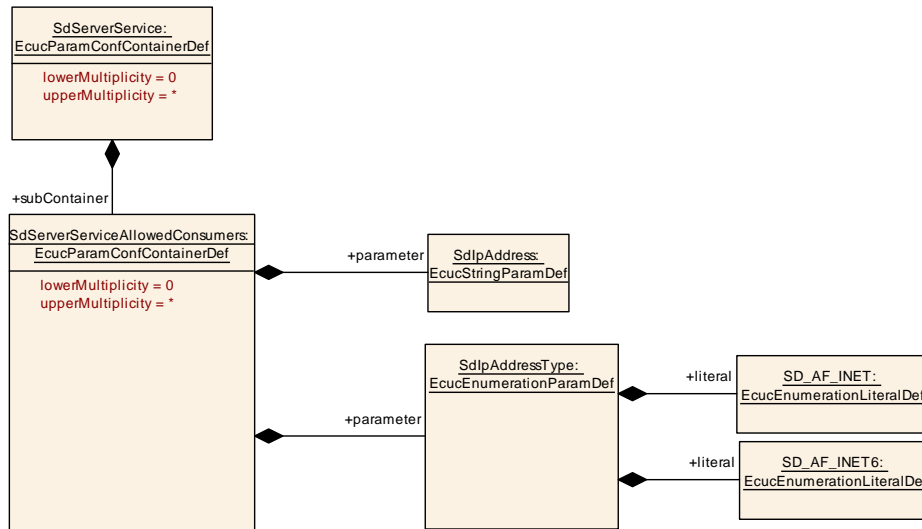


Figure 10.10: SdServerService Container2

### 10.2.18 SdEventHandler

SWS Item	[ECUC_SD_00055]		
Container Name	SdEventHandler		
Parent Container	<a href="#">SdServerService</a>		
Description	Container Element for representing an EventGroup as part of the Service Instance.		
Configuration Parameters			

SWS Item	[ECUC_SD_00061]		
Parameter Name	SdEventHandlerEventGroupId		
Parent Container	<a href="#">SdEventHandler</a>		
Description	The EventGroup Id of this EventGroup as a unique identifier of the EventGroup in this service. This identifier is used for EventGroup entries as well. Please note, that the Eventgroup ID 0x0000 is reserved.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65534		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00112]		
Parameter Name	SdEventHandlerHandleId		
Parent Container	<a href="#">SdEventHandler</a>		
Description	The HandleId by which the BswM can identify this EventGroup.		





<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU withAuto = true		

<b>SWS Item</b>	<b>[ECUC_SD_00097]</b>		
<b>Parameter Name</b>	SdEventHandlerMulticastThreshold		
<b>Parent Container</b>	<a href="#">SdEventHandler</a>		
<b>Description</b>	<p>Specifies the number of subscribed clients with different endpoint information (see SWS_SD_00754) that triggers the Server to change the transmission of events via the Eventhandler Multicast connection.</p> <p>If configured to 0 only Consumed Evengroup unicast connections and Consumed Eventgroup multicast connections will be used.</p> <p>If configured to 1 the first client and all further subscribed clients will be served via the Eventhandler Multicast connection as configured in SdMulticastEventSoConRef.</p> <p>If configured to n up to n-1 clients with different endpoint information will be served via Consumed Evengroup unicast connections and Consumed Eventgroup multicast connections. As soon as the number of subscribed clients with different endpoint information reaches n, then all subscribed clients are served via the Eventhandler Multicast connection as configured in SdMulticastEventSoConRef.</p> <p>This does not influence the handling of initial events.</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00113]</b>		
<b>Parameter Name</b>	SdEventHandlerTimerRef		
<b>Parent Container</b>	<a href="#">SdEventHandler</a>		
<b>Description</b>	The reference of the SdServerTimer container for this EventGroup.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Reference to <a href="#">SdServerTimer</a>		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE





	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
<a href="#">SdEventHandlerMulticast</a>	0..1	<p>The subcontainer including the Routing Group for Activation of Events sent over Multicast.</p> <p>The activation ref is also being used for identification of the related Socket Connection in order to find the Multicast Address used in the Multicast Option referenced by the Subscribe Event Group Ack entry.</p>
<a href="#">SdEventHandlerTcp</a>	0..1	<p>The subcontainer including the Routing Groups for Activation and Trigger Transmit for Events sent over TCP.</p> <p>The activation ref (or triggering ref if no activation ref exists) is also being used for identification of the related socket connections in order to find the related client by iterating the Sd EventHandlerTcp elements (remote address statically configured or automatically set by opening TCP connection before subscription).</p>
<a href="#">SdEventHandlerUdp</a>	0..1	<p>The subcontainer including the Routing Groups for Activation and Trigger Transmit for Events sent over UDP.</p> <p>The activation ref (or triggering ref if no activation ref exists) is also being used for identification of the related socket connections in order to set the remote address (either unicast address or multicast address) of the client or find the related client by iterating the SdEventHandlerUdp elements (remote address statically configured or automatically set by method call before subscription).</p>

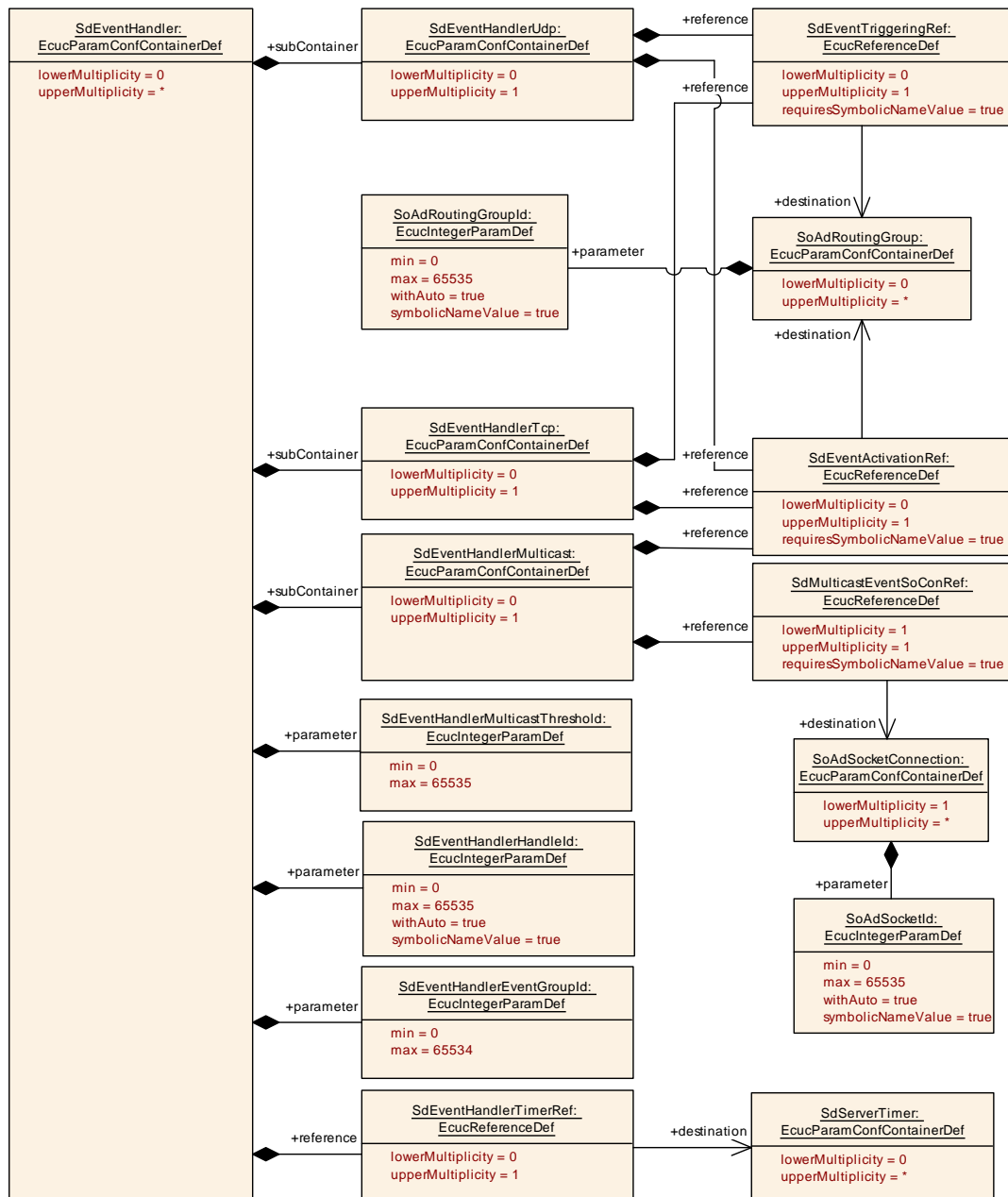


Figure 10.11: SdEventHandler Container

### 10.2.19 SdEventHandlerMulticast

SWS Item	[ECUC_SD_00094]
Container Name	SdEventHandlerMulticast
Parent Container	<a href="#">SdEventHandler</a>





<b>Description</b>	<p>The subcontainer including the Routing Group for Activation of Events sent over Multicast.</p> <p>The activation ref is also being used for identification of the related Socket Connection in order to find the Multicast Address used in the Multicast Option referenced by the Subscribe EventGroup Ack entry.</p>
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>[ECUC_SD_00096]</b>		
<b>Parameter Name</b>	SdEventActivationRef		
<b>Parent Container</b>	<a href="#">SdEventHandlerMulticast</a>		
<b>Description</b>	Reference to a SoAdRoutingGroup for activation of the data path for a subscribed client (start sending events after subscribe). This is usually equal to the SdEvent ActivationRef referenced by SdEventHandlerUdp		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>[ECUC_SD_00118]</b>		
<b>Parameter Name</b>	SdMulticastEventSoConRef		
<b>Parent Container</b>	<a href="#">SdEventHandlerMulticast</a>		
<b>Description</b>	Reference to the SoAdSocketConnection representing the Eventhandler Multicast data path (UDP).		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to SoAdSocketConnection		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

No Included Containers

## 10.2.20 SdEventHandlerTcp

<b>SWS Item</b>	[ECUC_SD_00093]
<b>Container Name</b>	SdEventHandlerTcp
<b>Parent Container</b>	<a href="#">SdEventHandler</a>
<b>Description</b>	<p>The subcontainer including the Routing Groups for Activation and Trigger Transmit for Events sent over TCP.</p> <p>The activation ref (or triggering ref if no activation ref exists) is also being used for identification of the related socket connections in order to find the related client by iterating the SdEventHandlerTcp elements (remote address statically configured or automatically set by opening TCP connection before subscription).</p>
<b>Configuration Parameters</b>	

<b>SWS Item</b>	[ECUC_SD_00096]		
<b>Parameter Name</b>	SdEventActivationRef		
<b>Parent Container</b>	<a href="#">SdEventHandlerTcp</a>		
<b>Description</b>	Reference to a SoAdRoutingGroup for activation of the data path for a subscribed client (start sending events after subscribe). This is usually equal to the SdEvent ActivationRef referenced by SdEventHandlerUdp		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	[ECUC_SD_00095]		
<b>Parameter Name</b>	SdEventTriggeringRef		
<b>Parent Container</b>	<a href="#">SdEventHandlerTcp</a>		
<b>Description</b>	Reference to a SoAdRoutingGroup that is used for triggered transmit. Triggering is needed to sent out initial events on the server side after a client got subscribed.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

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



## 10.2.21 SdEventHandlerUdp

<b>SWS Item</b>	[ECUC_SD_00092]
<b>Container Name</b>	SdEventHandlerUdp
<b>Parent Container</b>	<a href="#">SdEventHandler</a>
<b>Description</b>	<p>The subcontainer including the Routing Groups for Activation and Trigger Transmit for Events sent over UDP.</p> <p>The activation ref (or triggering ref if no activation ref exists) is also being used for identification of the related socket connections in order to set the remote address (either unicast address or multicast address) of the client or find the related client by iterating the SdEventHandlerUdp elements (remote address statically configured or automatically set by method call before subscription).</p>
<b>Configuration Parameters</b>	

<b>SWS Item</b>	[ECUC_SD_00096]		
<b>Parameter Name</b>	SdEventActivationRef		
<b>Parent Container</b>	<a href="#">SdEventHandlerUdp</a>		
<b>Description</b>	Reference to a SoAdRoutingGroup for activation of the data path for a subscribed client (start sending events after subscribe). This is usually equal to the SdEvent ActivationRef referenced by SdEventHandlerUdp		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	[ECUC_SD_00095]		
<b>Parameter Name</b>	SdEventTriggeringRef		
<b>Parent Container</b>	<a href="#">SdEventHandlerUdp</a>		
<b>Description</b>	Reference to a SoAdRoutingGroup that is used for triggered transmit. Triggering is needed to sent out initial events on the server side after a client got subscribed.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to SoAdRoutingGroup		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

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

## 10.2.22 SdProvidedMethods

SWS Item	[ECUC_SD_00087]
Container Name	SdProvidedMethods
Parent Container	<a href="#">SdServerService</a>
Description	Container element for representing the needed elements of the data path for the methods provided by the service.
Configuration Parameters	

SWS Item	[ECUC_SD_00090]		
Parameter Name	SdServerServiceActivationRef		
Parent Container	<a href="#">SdProvidedMethods</a>		
Description	Reference to a SoAdRoutingGroup to activate and deactivate the data path for methods of the service.		
Multiplicity	1		
Type	Symbolic name reference to SoAdRoutingGroup		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

## 10.2.23 SdServerCapabilityRecord

SWS Item	[ECUC_SD_00032]
Container Name	SdServerCapabilityRecord
Parent Container	<a href="#">SdServerService</a>
Description	<p>Sd uses capability records to store arbitrary name/value pairs conveying additional information about the named service.</p> <p>The following use cases are supported: 1) Key present, with no value (e.g. "passreq" – password required for this service)</p> <p>2) Key present, with empty value (e.g. "PlugIns=" server supports plugins, but none are presently installed)</p> <p>3) Key present, with non-empty value (e.g. "PlugIns=JPEG,MPEG2,MPEG4")</p>
Configuration Parameters	

SWS Item	[ECUC_SD_00033]
Parameter Name	SdServerCapabilityRecordKey
Parent Container	<a href="#">SdServerCapabilityRecord</a>
Description	Defines a CapabilityRecord key.
Multiplicity	1
Type	EcucStringParamDef
Default value	–
Regular Expression	–





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

SWS Item	[ECUC_SD_00034]		
Parameter Name	SdServerCapabilityRecordValue		
Parent Container	<a href="#">SdServerCapabilityRecord</a>		
Description	Defines the corresponding CapabilityRecord value.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

## 10.2.24 SdServerTimer

SWS Item	[ECUC_SD_00035]		
Container Name	SdServerTimer		
Parent Container	<a href="#">SdInstance</a>		
Description	This container specifies all timers used by the Service Discovery module for Server Services.		
Configuration Parameters			

SWS Item	[ECUC_SD_00039]		
Parameter Name	SdServerTimerInitialOfferDelayMax		
Parent Container	<a href="#">SdServerTimer</a>		
Description	Max value in [s] to delay randomly the first offer. This parameter is mandatory for ServerService.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		





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

SWS Item	[ECUC_SD_00038]		
Parameter Name	SdServerTimerInitialOfferDelayMin		
Parent Container	<a href="#">SdServerTimer</a>		
Description	Min value in [s] to delay randomly the first offer. This parameter is mandatory for Server Service.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_SD_00041]		
Parameter Name	SdServerTimerInitialOfferRepetitionBaseDelay		
Parent Container	<a href="#">SdServerTimer</a>		
Description	The base delay in [s] for offer repetitions. Successive offers have an exponential back off delay (1x base delay, 2x base delay, 4x base delay, ...). This parameter is mandatory for ServerService.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_SD_00040]		
Parameter Name	SdServerTimerInitialOfferRepetitionsMax		
Parent Container	<a href="#">SdServerTimer</a>		
Description	Configure the maximum amount of offer repetition. This parameter is mandatory for ServerService.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 10		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_SD_00076]		
Parameter Name	SdServerTimerOfferCyclicDelay		
Parent Container	<a href="#">SdServerTimer</a>		
Description	Interval between cyclic offers in the main phase. This parameter is mandatory for ServerService.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_SD_00114]		
Parameter Name	SdServerTimerRequestResponseMaxDelay		
Parent Container	<a href="#">SdServerTimer</a>		
Description	Maximum allowable response delay to entries received by multicast in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE





	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00115]</b>		
<b>Parameter Name</b>	SdServerTimerRequestResponseMinDelay		
<b>Parent Container</b>	<a href="#">SdServerTimer</a>		
<b>Description</b>	Minimum allowable response delay to entries received by multicast in seconds.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

<b>SWS Item</b>	<b>[ECUC_SD_00037]</b>		
<b>Parameter Name</b>	SdServerTimerTTL		
<b>Parent Container</b>	<a href="#">SdServerTimer</a>		
<b>Description</b>	Time to live for offer service.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 16777215		
<b>Default value</b>	–		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: ECU		

No Included Containers

### 10.2.25 SdServerServiceAllowedConsumers

<b>SWS Item</b>	<b>[ECUC_Sd_00155]</b>		
<b>Container Name</b>	SdServerServiceAllowedConsumers		
<b>Parent Container</b>	<a href="#">SdServerService</a>		
<b>Description</b>	This container defines a list of consumers that are allowed to access this SdServer Service. <b>Tags:</b> atp.Status=draft		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME





	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	[ECUC_Sd_00148]		
Parameter Name	SdIpAddress		
Parent Container	<a href="#">SdServerServiceAllowedConsumers</a>		
Description	This parameter defines the IP Address of the remote communication partner. <b>Tags:</b> atp.Status=draft		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

SWS Item	[ECUC_Sd_00149]		
Parameter Name	SdIpAddressType		
Parent Container	<a href="#">SdServerServiceAllowedConsumers</a>		
Description	This parameter defines the IP version that is used for communication with the remote communication partner. <b>Tags:</b> atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	SD_AF_INET	IPv4 address <b>Tags:</b> atp.Status=draft	
	SD_AF_INET6	IPv6 address <b>Tags:</b> atp.Status=draft	
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers
------------------------

## 10.2.26 SdClientServiceAllowedProvider

SWS Item	[ECUC_Sd_00147]
Container Name	SdClientServiceAllowedProvider
Parent Container	<a href="#">SdClientService</a>





<b>Description</b>	The container defines the allowed provider for this ClientService. <b>Tags:</b> atp.Status=draft		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>[ECUC_Sd_00148]</b>		
<b>Parameter Name</b>	SdIpAddress		
<b>Parent Container</b>	<a href="#">SdClientServiceAllowedProvider</a>		
<b>Description</b>	This parameter defines the IP Address of the remote communication partner. <b>Tags:</b> atp.Status=draft		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucStringParamDef		
<b>Default value</b>	–		
<b>Regular Expression</b>	–		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	–	
	<b>Post-build time</b>	–	
<b>Scope / Dependency</b>	scope: ECU		

SWS Item	[ECUC_Sd_00149]		
Parameter Name	SdIpAddressType		
Parent Container	SdClientServiceAllowedProvider		
Description	This parameter defines the IP version that is used for communication with the remote communication partner. <b>Tags:</b> atp.Status=draft		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	SD_AF_INET	IPv4 address <b>Tags:</b> atp.Status=draft	
	SD_AF_INET6	IPv6 address <b>Tags:</b> atp.Status=draft	
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: ECU		

No Included Containers



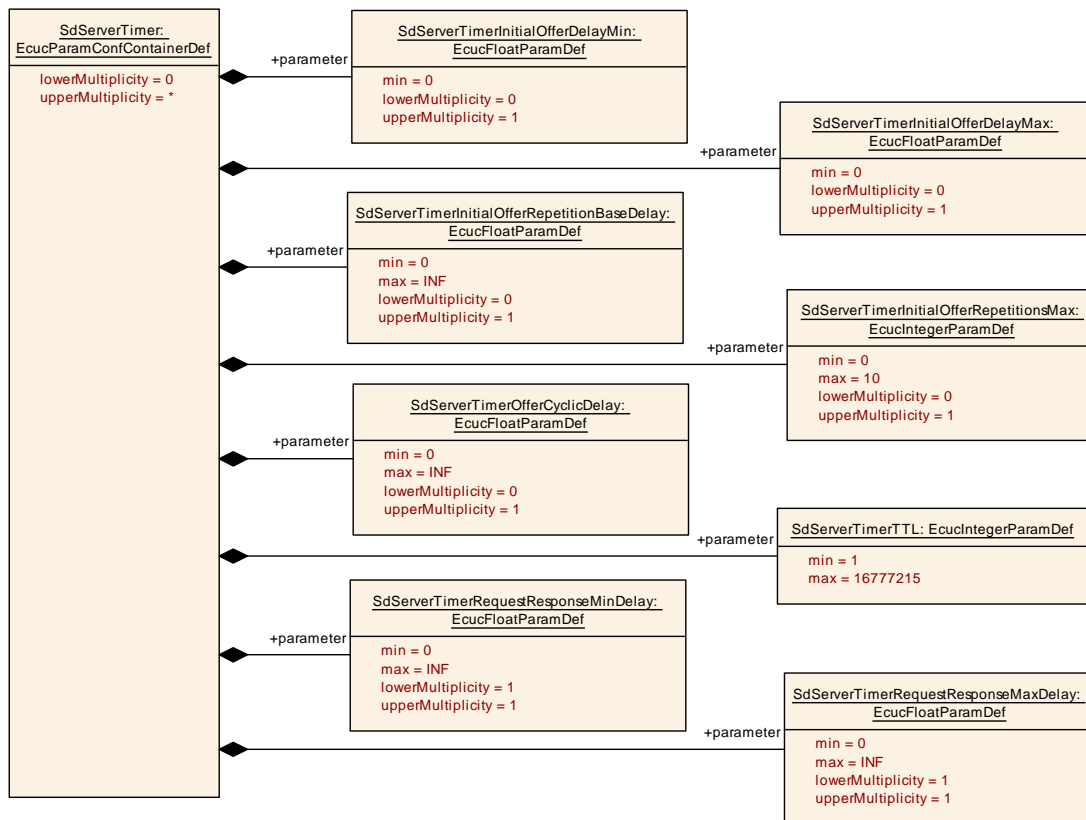


Figure 10.12: SdServerTimer Container

## 10.3 Published Information

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

## A 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.

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

#### A.1.1 Added Specification Items in R23-11

[\[SWS\\_Sd\\_00107\]](#) [\[SWS\\_Sd\\_00114\]](#) [\[SWS\\_Sd\\_00117\]](#) [\[SWS\\_Sd\\_00118\]](#) [\[SWS\\_Sd\\_00119\]](#) [\[SWS\\_Sd\\_00124\]](#) [\[SWS\\_Sd\\_00129\]](#) [\[SWS\\_Sd\\_00130\]](#) [\[SWS\\_Sd\\_00133\]](#) [\[SWS\\_Sd\\_00134\]](#) [\[SWS\\_Sd\\_00405\]](#) [\[SWS\\_Sd\\_00409\]](#) [\[SWS\\_Sd\\_00412\]](#) [\[SWS\\_Sd\\_00496\]](#) [\[SWS\\_Sd\\_00550\]](#) [\[SWS\\_Sd\\_00551\]](#) [\[SWS\\_Sd\\_00552\]](#) [\[SWS\\_Sd\\_00553\]](#) [\[SWS\\_Sd\\_00560\]](#) [\[SWS\\_Sd\\_00690\]](#) [\[SWS\\_Sd\\_00742\]](#) [\[SWS\\_Sd\\_00762\]](#) [\[SWS\\_Sd\\_00763\]](#) [\[SWS\\_Sd\\_00764\]](#) [\[SWS\\_Sd\\_00765\]](#) [\[SWS\\_Sd\\_00766\]](#) [\[SWS\\_Sd\\_00767\]](#) [\[SWS\\_Sd\\_00768\]](#) [\[SWS\\_Sd\\_00769\]](#) [\[SWS\\_Sd\\_00780\]](#) [\[SWS\\_Sd\\_00781\]](#) [\[SWS\\_Sd\\_00782\]](#) [\[SWS\\_Sd\\_00783\]](#) [\[SWS\\_Sd\\_00784\]](#) [\[SWS\\_Sd\\_00785\]](#) [\[SWS\\_Sd\\_00786\]](#) [\[SWS\\_Sd\\_00787\]](#) [\[SWS\\_Sd\\_00788\]](#) [\[SWS\\_Sd\\_00789\]](#) [\[SWS\\_Sd\\_00790\]](#) [\[SWS\\_Sd\\_00791\]](#) [\[SWS\\_Sd\\_00792\]](#) [\[SWS\\_Sd\\_00793\]](#) [\[SWS\\_Sd\\_00794\]](#) [\[SWS\\_Sd\\_00795\]](#) [\[SWS\\_Sd\\_00796\]](#) [\[SWS\\_Sd\\_00797\]](#) [\[SWS\\_Sd\\_00798\]](#) [\[SWS\\_Sd\\_00799\]](#) [\[SWS\\_Sd\\_00800\]](#) [\[SWS\\_Sd\\_91001\]](#) [\[SWS\\_Sd\\_91002\]](#) [\[SWS\\_Sd\\_91003\]](#) [\[SWS\\_Sd\\_91006\]](#) [\[SWS\\_Sd\\_91007\]](#) [\[SWS\\_Sd\\_91008\]](#) [\[SWS\\_Sd\\_91009\]](#) [\[SWS\\_Sd\\_91010\]](#) [\[SWS\\_Sd\\_91011\]](#)

#### A.1.2 Changed Specification Items in R23-11

[\[SWS\\_Sd\\_00352\]](#) [\[SWS\\_Sd\\_00365\]](#) [\[SWS\\_Sd\\_00448\]](#) [\[SWS\\_Sd\\_00453\]](#) [\[SWS\\_Sd\\_00464\]](#) [\[SWS\\_Sd\\_00488\]](#) [\[SWS\\_Sd\\_00489\]](#) [\[SWS\\_Sd\\_00721\]](#) [\[SWS\\_Sd\\_00730\]](#) [\[SWS\\_Sd\\_00731\]](#) [\[SWS\\_Sd\\_00732\]](#) [\[SWS\\_Sd\\_00733\]](#) [\[SWS\\_Sd\\_00737\]](#)

#### A.1.3 Deleted Specification Items in R23-11

[\[SWS\\_SD\\_00107\]](#) [\[SWS\\_SD\\_00117\]](#) [\[SWS\\_SD\\_00118\]](#) [\[SWS\\_SD\\_00119\]](#) [\[SWS\\_SD\\_00124\]](#) [\[SWS\\_SD\\_00129\]](#) [\[SWS\\_SD\\_00130\]](#) [\[SWS\\_SD\\_00133\]](#) [\[SWS\\_SD\\_00134\]](#) [\[SWS\\_SD\\_00405\]](#) [\[SWS\\_SD\\_00409\]](#) [\[SWS\\_SD\\_00412\]](#) [\[SWS\\_SD\\_00496\]](#) [\[SWS\\_SD\\_00550\]](#) [\[SWS\\_SD\\_00551\]](#) [\[SWS\\_SD\\_00552\]](#) [\[SWS\\_SD\\_00553\]](#) [\[SWS\\_SD\\_00560\]](#) [\[SWS\\_SD\\_00690\]](#) [\[SWS\\_SD\\_00742\]](#) [\[SWS\\_SD\\_91001\]](#) [\[SWS\\_SD\\_91002\]](#) [\[SWS\\_SD\\_91003\]](#) [\[SWS\\_SD\\_91006\]](#) [\[SWS\\_SD\\_91007\]](#) [\[SWS\\_SD\\_91008\]](#) [\[SWS\\_Sd\\_00037\]](#) [\[SWS\\_Sd\\_00319\]](#) [\[SWS\\_Sd\\_00376\]](#) [\[SWS\\_Sd\\_00604\]](#)