Document Title	SOME/IP Service Discovery Protocol Specification
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	802

Document Status	published
Part of AUTOSAR Standard	Foundation
Part of Standard Release	R19-11

Document Change History

Date	Release	Changed by	Description
2019-11-28	R19-11	AUTOSAR Release Management	• Clarify: – Startup Behavior (random value) – Service Versioning in VLAN – Load Balancing option behavior – Re-boot Detection • Introduce retry max counter for subscription of Eventgroup • Contradicting requirements improved • Editorial changes • Changed Document Status from Final to published
2019-03-29	1.5.1	AUTOSAR Release Management	• Editorial changes
2018-10-31	1.5.0	AUTOSAR Release Management	• Clarify load balancing option usage • Contradicting requirements improved • Redundant requirements removed
2018-03-29	1.4.0	AUTOSAR Release Management	• No content changes
2017-12-08	1.3.0	AUTOSAR Release Management	• minor changes
2017-10-27	1.2.0	AUTOSAR Release Management	• Editorial changes

2017-03-31	1.1.0	AUTOSAR Release Management	• Configuration Parameters SD_PORT and SD_MULTICAST_IP are added and defined • Rules relating to Options are reordered
2016-11-30	1.0.0	AUTOSAR Release Management	Initial Release

Disclaimer

Table of Contents

1 Introduction and overview

This protocol specification specifies the format, message sequences and semantics of the Protocol SOME/IP Service Discovery (SOME/IP-SD).

The main tasks of the Service Discovery Protocol are communicating 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).

1.1 Protocol purpose and objectives

SOME/IP-SD is used to

- Locate service instances.
- Detect if service instances are running.
- Implement the Publish/Subscribe handling.

1.2 Applicability of the protocol

SOME/IP SD can be used for service discovery in automotive vehicle networks.

1.2.1 Constraints and assumptions

Currently SOME/IP-SD supports only IP based communication.

1.3 Dependencies

1.3.1 Dependencies to other protocol layers

SOME/IP-SD depends on SOME/IP. SOME/IP itself supports both TCP and UDP communications but SOME/IP SD is constraint to use SOME/IP only over UDP (See [PRS_SOMEIPSD_00220]).

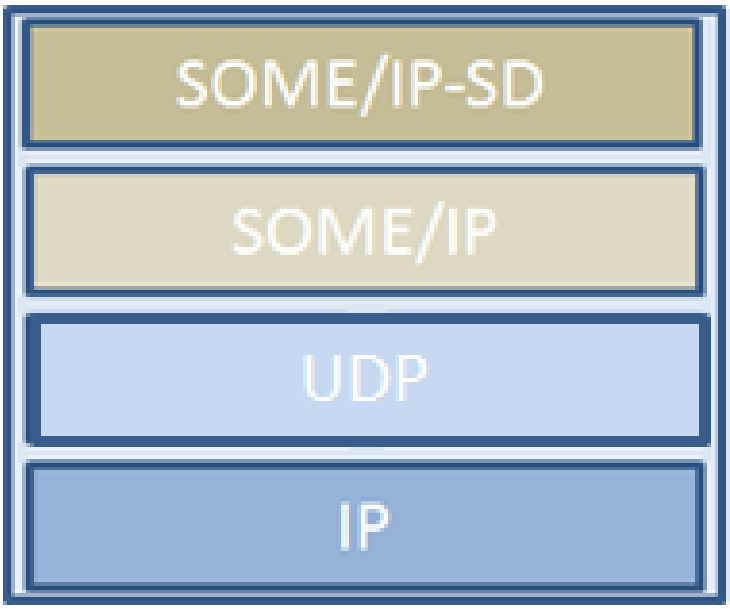

Figure 1.1: SOME/IP-SD Dependencies to other protocol layers

2 Protocol Requirements

2.1 Requirements Traceability

Feature	Description	Satisfied by
[RS_SOMEIPSD_00001]	SOME/IP Service Discovery Protocol shall be used on top of SOME/IP Protocol	[PRS_SOMEIPSD_00151] [PRS_SOMEIPSD_00152] [PRS_SOMEIPSD_00153] [PRS_SOMEIPSD_00154] [PRS_SOMEIPSD_00155] [PRS_SOMEIPSD_00156] [PRS_SOMEIPSD_00157] [PRS_SOMEIPSD_00158] [PRS_SOMEIPSD_00159] [PRS_SOMEIPSD_00160] [PRS_SOMEIPSD_00161] [PRS_SOMEIPSD_00162] [PRS_SOMEIPSD_00163] [PRS_SOMEIPSD_00164] [PRS_SOMEIPSD_00250] [PRS_SOMEIPSD_00251] [PRS_SOMEIPSD_00252] [PRS_SOMEIPSD_00600]
[RS_SOMEIPSD_00002]	SOME/IP Service Discovery Protocol shall support unicast messages	[PRS_SOMEIPSD_00256] [PRS_SOMEIPSD_00259] [PRS_SOMEIPSD_00540] [PRS_SOMEIPSD_00601] [PRS_SOMEIPSD_00602] [PRS_SOMEIPSD_00631] [PRS_SOMEIPSD_00700] [PRS_SOMEIPSD_00701] [PRS_SOMEIPSD_00702]
[RS_SOMEIPSD_00003]	SOME/IP Service Discovery Protocol shall support multicast messages	[PRS_SOMEIPSD_00238] [PRS_SOMEIPSD_00239] [PRS_SOMEIPSD_00256] [PRS_SOMEIPSD_00323] [PRS_SOMEIPSD_00324] [PRS_SOMEIPSD_00325] [PRS_SOMEIPSD_00326] [PRS_SOMEIPSD_00329] [PRS_SOMEIPSD_00331] [PRS_SOMEIPSD_00332] [PRS_SOMEIPSD_00333] [PRS_SOMEIPSD_00336] [PRS_SOMEIPSD_00545] [PRS_SOMEIPSD_00601] [PRS_SOMEIPSD_00603] [PRS_SOMEIPSD_00631]
[RS_SOMEIPSD_00004]	SOME/IP Service Discovery Protocol shall support SOME/IP and non-SOME/IP services	[PRS_SOMEIPSD_00437] [PRS_SOMEIPSD_00438] [PRS_SOMEIPSD_00439] [PRS_SOMEIPSD_00440]

[RS_SOMEIPSD_00005]	SOME/IP Service Discovery Protocol shall support different versions of the same service	[PRS_SOMEIPSD_00512] [PRS_SOMEIPSD_00806]
[RS_SOMEIPSD_00006]	SOME/IP Service Discovery Protocol shall define the format of the Service Discovery message	[PRS_SOMEIPSD_00253] [PRS_SOMEIPSD_00254] [PRS_SOMEIPSD_00255] [PRS_SOMEIPSD_00258] [PRS_SOMEIPSD_00261] [PRS_SOMEIPSD_00262] [PRS_SOMEIPSD_00263] [PRS_SOMEIPSD_00264] [PRS_SOMEIPSD_00265] [PRS_SOMEIPSD_00266] [PRS_SOMEIPSD_00267] [PRS_SOMEIPSD_00268] [PRS_SOMEIPSD_00269] [PRS_SOMEIPSD_00270] [PRS_SOMEIPSD_00273] [PRS_SOMEIPSD_00274] [PRS_SOMEIPSD_00275] [PRS_SOMEIPSD_00276] [PRS_SOMEIPSD_00277] [PRS_SOMEIPSD_00278] [PRS_SOMEIPSD_00279] [PRS_SOMEIPSD_00280] [PRS_SOMEIPSD_00281] [PRS_SOMEIPSD_00282] [PRS_SOMEIPSD_00283] [PRS_SOMEIPSD_00284] [PRS_SOMEIPSD_00285] [PRS_SOMEIPSD_00286] [PRS_SOMEIPSD_00287] [PRS_SOMEIPSD_00289] [PRS_SOMEIPSD_00305] [PRS_SOMEIPSD_00306] [PRS_SOMEIPSD_00307] [PRS_SOMEIPSD_00310] [PRS_SOMEIPSD_00314] [PRS_SOMEIPSD_00315] [PRS_SOMEIPSD_00319] [PRS_SOMEIPSD_00320] [PRS_SOMEIPSD_00321] [PRS_SOMEIPSD_00380] [PRS_SOMEIPSD_00547] [PRS_SOMEIPSD_00548] [PRS_SOMEIPSD_00549] [PRS_SOMEIPSD_00550] [PRS_SOMEIPSD_00551] [PRS_SOMEIPSD_00552] [PRS_SOMEIPSD_00554] [PRS_SOMEIPSD_00555]

		[PRS_SOMEIPSD_00556] [PRS_SOMEIPSD_00557] [PRS_SOMEIPSD_00558] [PRS_SOMEIPSD_00559] [PRS_SOMEIPSD_00650] [PRS_SOMEIPSD_00651] [PRS_SOMEIPSD_00654] [PRS_SOMEIPSD_00807]
[RS_SOMEIPSD_00007]	SOME/IP Service Discovery Protocol shall define ordered feature sets for compliance of implementations	[PRS_SOMEIPSD_00496] [PRS_SOMEIPSD_00497] [PRS_SOMEIPSD_00498] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00501] [PRS_SOMEIPSD_00502] [PRS_SOMEIPSD_00503] [PRS_SOMEIPSD_00504] [PRS_SOMEIPSD_00821]
[RS_SOMEIPSD_00008]	SOME/IP Service Discovery Protocol shall support to find the location of service instances	[PRS_SOMEIPSD_00350] [PRS_SOMEIPSD_00351] [PRS_SOMEIPSD_00496] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00501] [PRS_SOMEIPSD_00512] [PRS_SOMEIPSD_00528] [PRS_SOMEIPSD_00583] [PRS_SOMEIPSD_00806]
[RS_SOMEIPSD_00009]	SOME/IP Service Discovery Protocol shall support to transport text-based names of services	[PRS_SOMEIPSD_00277]
[RS_SOMEIPSD_00010]	SOME/IP Service Discovery Protocol shall provide support to transport optional data	[PRS_SOMEIPSD_00220] [PRS_SOMEIPSD_00305] [PRS_SOMEIPSD_00306] [PRS_SOMEIPSD_00307] [PRS_SOMEIPSD_00310] [PRS_SOMEIPSD_00314] [PRS_SOMEIPSD_00315] [PRS_SOMEIPSD_00319] [PRS_SOMEIPSD_00320] [PRS_SOMEIPSD_00321] [PRS_SOMEIPSD_00380] [PRS_SOMEIPSD_00547] [PRS_SOMEIPSD_00548] [PRS_SOMEIPSD_00549] [PRS_SOMEIPSD_00550] [PRS_SOMEIPSD_00551] [PRS_SOMEIPSD_00552] [PRS_SOMEIPSD_00554] [PRS_SOMEIPSD_00555] [PRS_SOMEIPSD_00556] [PRS_SOMEIPSD_00557] [PRS_SOMEIPSD_00558] [PRS_SOMEIPSD_00559] [PRS_SOMEIPSD_00650]

		[PRS_SOMEIPSD_00651] [PRS_SOMEIPSD_00654] [PRS_SOMEIPSD_00807]
[RS_SOMEIPSD_00011]	SOME/IP Service Discovery Protocol shall provide support for load balancing	[PRS_SOMEIPSD_00542] [PRS_SOMEIPSD_00544] [PRS_SOMEIPSD_00711] [PRS_SOMEIPSD_00712] [PRS_SOMEIPSD_00713] [PRS_SOMEIPSD_00714]
[RS_SOMEIPSD_00012]	SOME/IP Service Discovery Protocol shall support to detect whether service instances are active	[PRS_SOMEIPSD_00133] [PRS_SOMEIPSD_00397] [PRS_SOMEIPSD_00427]
[RS_SOMEIPSD_00013]	SOME/IP Service Discovery Protocol shall support to offer published services	[PRS_SOMEIPSD_00355] [PRS_SOMEIPSD_00356] [PRS_SOMEIPSD_00357] [PRS_SOMEIPSD_00358] [PRS_SOMEIPSD_00359] [PRS_SOMEIPSD_00360] [PRS_SOMEIPSD_00361] [PRS_SOMEIPSD_00362] [PRS_SOMEIPSD_00443] [PRS_SOMEIPSD_00446] [PRS_SOMEIPSD_00457] [PRS_SOMEIPSD_00480] [PRS_SOMEIPSD_00481] [PRS_SOMEIPSD_00496] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00504] [PRS_SOMEIPSD_00512] [PRS_SOMEIPSD_00529] [PRS_SOMEIPSD_00530] [PRS_SOMEIPSD_00583] [PRS_SOMEIPSD_00801] [PRS_SOMEIPSD_00802] [PRS_SOMEIPSD_00806] [PRS_SOMEIPSD_00821]
[RS_SOMEIPSD_00014]	SOME/IP Service Discovery Protocol shall support to stop offering services	[PRS_SOMEIPSD_00363] [PRS_SOMEIPSD_00364] [PRS_SOMEIPSD_00443] [PRS_SOMEIPSD_00446] [PRS_SOMEIPSD_00496] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00583]

[RS_SOMEIPSD_00015]	SOME/IP Service Discovery Protocol shall support to subscribe to events	[PRS_SOMEIPSD_00120] [PRS_SOMEIPSD_00121] [PRS_SOMEIPSD_00122] [PRS_SOMEIPSD_00123] [PRS_SOMEIPSD_00385] [PRS_SOMEIPSD_00386] [PRS_SOMEIPSD_00387] [PRS_SOMEIPSD_00390] [PRS_SOMEIPSD_00391] [PRS_SOMEIPSD_00392] [PRS_SOMEIPSD_00393] [PRS_SOMEIPSD_00394] [PRS_SOMEIPSD_00443] [PRS_SOMEIPSD_00446] [PRS_SOMEIPSD_00449] [PRS_SOMEIPSD_00450] [PRS_SOMEIPSD_00453] [PRS_SOMEIPSD_00457] [PRS_SOMEIPSD_00461] [PRS_SOMEIPSD_00462] [PRS_SOMEIPSD_00463] [PRS_SOMEIPSD_00464] [PRS_SOMEIPSD_00465] [PRS_SOMEIPSD_00466] [PRS_SOMEIPSD_00467] [PRS_SOMEIPSD_00468] [PRS_SOMEIPSD_00470] [PRS_SOMEIPSD_00472] [PRS_SOMEIPSD_00484] [PRS_SOMEIPSD_00486] [PRS_SOMEIPSD_00487] [PRS_SOMEIPSD_00488] [PRS_SOMEIPSD_00489] [PRS_SOMEIPSD_00490] [PRS_SOMEIPSD_00496] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00501] [PRS_SOMEIPSD_00504] [PRS_SOMEIPSD_00512] [PRS_SOMEIPSD_00527] [PRS_SOMEIPSD_00566] [PRS_SOMEIPSD_00570] [PRS_SOMEIPSD_00571] [PRS_SOMEIPSD_00572] [PRS_SOMEIPSD_00577] [PRS_SOMEIPSD_00583] [PRS_SOMEIPSD_00703] [PRS_SOMEIPSD_00704] [PRS_SOMEIPSD_00806] [PRS_SOMEIPSD_00808] [PRS_SOMEIPSD_00809] [PRS_SOMEIPSD_00821] [PRS_SOMEIPSD_00822] [PRS_SOMEIPSD_00824]

[RS_SOMEIPSD_00016]	SOME/IP Service Discovery Protocol shall support to deny subscriptions	[PRS_SOMEIPSD_00134] [PRS_SOMEIPSD_00443] [PRS_SOMEIPSD_00446] [PRS_SOMEIPSD_00466] [PRS_SOMEIPSD_00467] [PRS_SOMEIPSD_00468] [PRS_SOMEIPSD_00583]
[RS_SOMEIPSD_00017]	SOME/IP Service Discovery Protocol shall support to stop subscriptions to events	[PRS_SOMEIPSD_00388] [PRS_SOMEIPSD_00389] [PRS_SOMEIPSD_00427] [PRS_SOMEIPSD_00428] [PRS_SOMEIPSD_00429] [PRS_SOMEIPSD_00430] [PRS_SOMEIPSD_00431] [PRS_SOMEIPSD_00432] [PRS_SOMEIPSD_00452] [PRS_SOMEIPSD_00453] [PRS_SOMEIPSD_00454] [PRS_SOMEIPSD_00496] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00574] [PRS_SOMEIPSD_00751] [PRS_SOMEIPSD_00752]
[RS_SOMEIPSD_00018]	SOME/IP Service Discovery Protocol shall support reboot detection of service providers	[PRS_SOMEIPSD_00503]
[RS_SOMEIPSD_00019]	SOME/IP Service Discovery Protocol shall standardize error handling	[PRS_SOMEIPSD_00125] [PRS_SOMEIPSD_00126] [PRS_SOMEIPSD_00127] [PRS_SOMEIPSD_00128] [PRS_SOMEIPSD_00129] [PRS_SOMEIPSD_00130] [PRS_SOMEIPSD_00131] [PRS_SOMEIPSD_00132] [PRS_SOMEIPSD_00231] [PRS_SOMEIPSD_00232] [PRS_SOMEIPSD_00233] [PRS_SOMEIPSD_00234] [PRS_SOMEIPSD_00235] [PRS_SOMEIPSD_00454] [PRS_SOMEIPSD_00803]
[RS_SOMEIPSD_00020]	SOME/IP Service Discovery Protocol shall support TTL	[PRS_SOMEIPSD_00452] [PRS_SOMEIPSD_00502] [PRS_SOMEIPSD_00704]
[RS_SOMEIPSD_00021]	SOME/IP Service Discovery protocol shall provide functionality to discover services	[PRS_SOMEIPSD_00350] [PRS_SOMEIPSD_00351]
[RS_SOMEIPSD_00022]	SOME/IP Service Discovery shall operate in a distributed manner	[PRS_SOMEIPSD_00603]
[RS_SOMEIPSD_00024]	SOME/IP Service Discovery shall support configurable timings	[PRS_SOMEIPSD_00502]

[RS_SOMEIPSD_00025]	SOME/IP Service Discovery messages shall contain information how to contact the communication partner	[PRS_SOMEIPSD_00133] [PRS_SOMEIPSD_00134] [PRS_SOMEIPSD_00341] [PRS_SOMEIPSD_00342] [PRS_SOMEIPSD_00343] [PRS_SOMEIPSD_00356] [PRS_SOMEIPSD_00357] [PRS_SOMEIPSD_00358] [PRS_SOMEIPSD_00359] [PRS_SOMEIPSD_00360] [PRS_SOMEIPSD_00361] [PRS_SOMEIPSD_00362] [PRS_SOMEIPSD_00387] [PRS_SOMEIPSD_00395] [PRS_SOMEIPSD_00397] [PRS_SOMEIPSD_00399] [PRS_SOMEIPSD_00400] [PRS_SOMEIPSD_00401] [PRS_SOMEIPSD_00404] [PRS_SOMEIPSD_00405] [PRS_SOMEIPSD_00406] [PRS_SOMEIPSD_00407] [PRS_SOMEIPSD_00408] [PRS_SOMEIPSD_00409] [PRS_SOMEIPSD_00410] [PRS_SOMEIPSD_00411] [PRS_SOMEIPSD_00412] [PRS_SOMEIPSD_00413] [PRS_SOMEIPSD_00415] [PRS_SOMEIPSD_00416] [PRS_SOMEIPSD_00417] [PRS_SOMEIPSD_00419] [PRS_SOMEIPSD_00420] [PRS_SOMEIPSD_00421] [PRS_SOMEIPSD_00422] [PRS_SOMEIPSD_00423] [PRS_SOMEIPSD_00433] [PRS_SOMEIPSD_00434] [PRS_SOMEIPSD_00435] [PRS_SOMEIPSD_00470] [PRS_SOMEIPSD_00476] [PRS_SOMEIPSD_00480] [PRS_SOMEIPSD_00481] [PRS_SOMEIPSD_00484] [PRS_SOMEIPSD_00486] [PRS_SOMEIPSD_00487] [PRS_SOMEIPSD_00488] [PRS_SOMEIPSD_00489]

		[PRS_SOMEIPSD_00490] [PRS_SOMEIPSD_00497] [PRS_SOMEIPSD_00498] [PRS_SOMEIPSD_00500] [PRS_SOMEIPSD_00501] [PRS_SOMEIPSD_00502] [PRS_SOMEIPSD_00504] [PRS_SOMEIPSD_00512] [PRS_SOMEIPSD_00515] [PRS_SOMEIPSD_00516] [PRS_SOMEIPSD_00517] [PRS_SOMEIPSD_00519] [PRS_SOMEIPSD_00520] [PRS_SOMEIPSD_00528] [PRS_SOMEIPSD_00529] [PRS_SOMEIPSD_00530] [PRS_SOMEIPSD_00531] [PRS_SOMEIPSD_00582] [PRS_SOMEIPSD_00583] [PRS_SOMEIPSD_00656] [PRS_SOMEIPSD_00800] [PRS_SOMEIPSD_00801] [PRS_SOMEIPSD_00802] [PRS_SOMEIPSD_00804] [PRS_SOMEIPSD_00805] [PRS_SOMEIPSD_00806] [PRS_SOMEIPSD_00821]

3 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the SOME/IP specification that are not included in the [1, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
Method	a method, procedure, function, or subroutine that is called/invoked.
Parameters	input, output, or input/output arguments of a method or an event
Remote Procedure Call (RPC)	a method call from one ECU to another that is transmitted using messages
Request	a message of the client to the server invoking a method
Response	a message of the server to the client transporting results of a method invocation
Request/Response communication	a RPC that consists of request and response
Fire&Forget communication	a RPC call that consists only of a request message
Event	A uni-directional data transmission that is only invoked on changes or cyclically and is sent from the producer of data to the consumers. One event could even be both sent cyclically and spontaneously on change. This is completely in the responsibility of the sending application because the receiver has no means at all to distinguish between those two.
Field	a field does represent a status and thus has a valid value at all times on which getter, setter and notfier act upon.
Notification Event	an event message the notifier of a field sends. The message of such a notifier cannot be distinguished from the event message; therefore, when refering to the message of an event, this shall also be true for the messages of notifiers of fields.
Getter	a Request/Response call that allows read access to a field.
Setter	a Request/Response call that allows write access to a field.
Notifier	sends out event message with a new value on change of the value of the field.
Service	a logical combination of zero or more methods, zero or more events, and zero or more fields (empty service is allowed, e.g. for announcing non-SOME/IP services in SOME/IP-SD)
Eventgroup	a logical grouping of events and notification events of fields inside a service in order to allow subscription
Service Interface	the formal specification of the service including its methods, events, and fields
Service Instance	software implementation of the service interface, which can exist more than once in the vehicle and more than once on an ECU
Server	The ECU offering a service instance shall be called server in the context of this service instance.
Client	The ECU using the service instance of a server shall be called client in the context of this service instance.
Union or Variant	a data structure that dynamically assumes different data types.
Offering a service instance	that one ECU implements an instance of a service and tells other ECUs using SOME/IP-SD that they may use it.
Finding a service instance	to send a SOME/IP-SD message in order to find a needed service instance.

Abbreviation / Acronym:	Description:
Requiring a service instance	to send a SOME/IP-SD message to the ECU implementing the required service instance with the meaning that this service instance is needed by the other ECU. This may be also sent if the service instance is not running; thus, was not offered yet.
Releasing a service instance	to sent a SOME/IP-SD message to the ECU hosting this service instances with the meaning that the service instance is no longer needed.
Server-Service-Instance-Entry	The configuration and required data of a service instance the local ECU offers, is called Server-Service-Instance-Entry at the ECU offering this service (Server).
Client-Service-Instance-Entry	The configuration and required data of a service instance another ECU offers, is called Client-Service-Instance-Entry at the ECU using this service (Client)
Publishing an eventgroup	to offer an eventgroup of a service instance to other ECUs using a SOME/IP-SD message
Subscribing an eventgroup	to require an eventgroup of a service instance using a SOME/IP-SD message

Table 3.1: Acronyms and Abbreviations

4 Protocol specification

4.1 SOME/IP Service Discovery (SOME/IP-SD)

4.1.1 General

SOME/IP-SD is used to

- Locate service instances.
- Detect if service instances are running.
- Implement the Publish/Subscribe handling.

Inside the vehicular network service instance locations are commonly known; therefore, the state of the service instance is of primary concern. The location of the service (i.e. IP-Address, transport protocol, and port number) are of secondary concern.

4.1.1.1 Terms and Definitions

[PRS_SOMEIPSD_00238] ⌈A separate server service instance shall be used per interface if a service instance needs to be offered on multiple interfaces.⌋*(RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00239] ⌈A separate client service instance shall be used per interface if a service instance needs to be configured to be accessed using multiple different interfaces.⌋*(RS_SOMEIPSD_00003)*

4.1.2 SOME/IP-SD Message Format

4.1.2.1 General Requirements

[PRS_SOMEIPSD_00220] ⌈SOME/IP-SD messages shall be sent over UDP.⌋*(RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00251] ⌈The SOME/IP-SD Header Format shall follow:

- Message ID (Service ID/Method ID) [32 bit]: `0xFFFF 8100`
- Length [32 bit]
- Request ID (Client ID/Session ID) [32 bit]
- Protocol Version [8 bit]: `0x01`
- Interface Version [8 bit]: `0x01`
- Message Type [8 bit]: `0x02`

- Return Code [8 bit]: 0x00
- Flags [8 bit]
- Reserved [24 bit]
- Length of Entries Array [32 bit]
- Entries Array [variable size]
- Length of Options Array [32 bit]
- Options Array [variable size]

⌋(*RS_SOMEIPSD_00001*)

The SOME/IP-SD Header Format is shown in Figure 4.1.

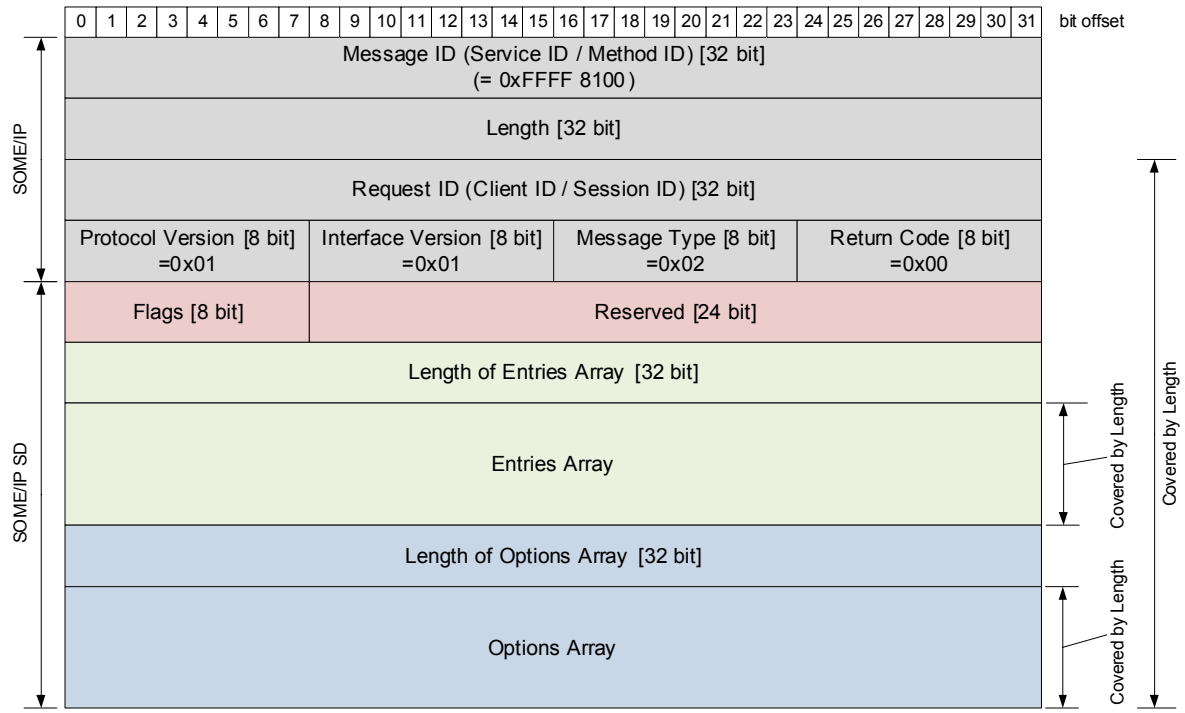

Figure 4.1: SOME/IP-SD Header Format

[PRS_SOMEIPSD_00250] ⌈Service Discovery Messages shall start with a SOME/IP header.⌋(*RS_SOMEIPSD_00001*)

[PRS_SOMEIPSD_00250] can be seen in Figure 4.1.

[PRS_SOMEIPSD_00151] ⌈Service Discovery messages shall use the Service-ID (16 Bits) of 0xFFFF.⌋(*RS_SOMEIPSD_00001*)

[PRS_SOMEIPSD_00152] ⌈Service Discovery messages shall use the Method-ID (16 Bits) of 0x8100.⌋(*RS_SOMEIPSD_00001*)

[PRS_SOMEIPSD_00153] ⌈Service Discovery messages shall use a uint32 length field as specified by SOME/IP. That means that the length is measured in bytes and starts with the first byte after the length field and ends with the last byte of the SOME/IP-SD message.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00154] ⌈Service Discovery messages shall have a Client-ID (16 Bits) set to 0x0000, since there exists only a single SOME/IP-SD instance.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00155] ⌈If a Client ID Prefix is configured, it shall also apply to SOME/IP-SD (See [PRS_SOMEIP_00532] in [2]).⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00156] ⌈Service Discovery messages shall have a Session-ID (16 Bits) and handle it based on SOME/IP requirements.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00157] ⌈The Session-ID (SOME/IP header) shall be incremented for every SOME/IP-SD message sent.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00158] ⌈The Session-ID (SOME/IP header) shall start with 1 and be 1 even after wrapping.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00159] ⌈The Session-ID (SOME/IP header) shall not be set to 0.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00160] ⌈SOME/IP-SD Session ID handling is done per "communication relation", i.e. broadcast as well as unicast per peer (see [PRS_SOMEIPSD_00255]).⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00161] ⌈Service Discovery messages shall have a Protocol-Version (8 Bits) of 0x01.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00162] ⌈Service Discovery messages shall have a Interface-Version (8 Bits) of 0x01.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00163] ⌈Service Discovery messages shall have a Message Type (8 bits) of 0x02 (Notification).⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00164] ⌈Service Discovery messages shall have a Return Code (8 bits) of 0x00 (E_OK).⌋*(RS_SOMEIPSD_00001)*

An example of SOME/IP-SD is as shown below

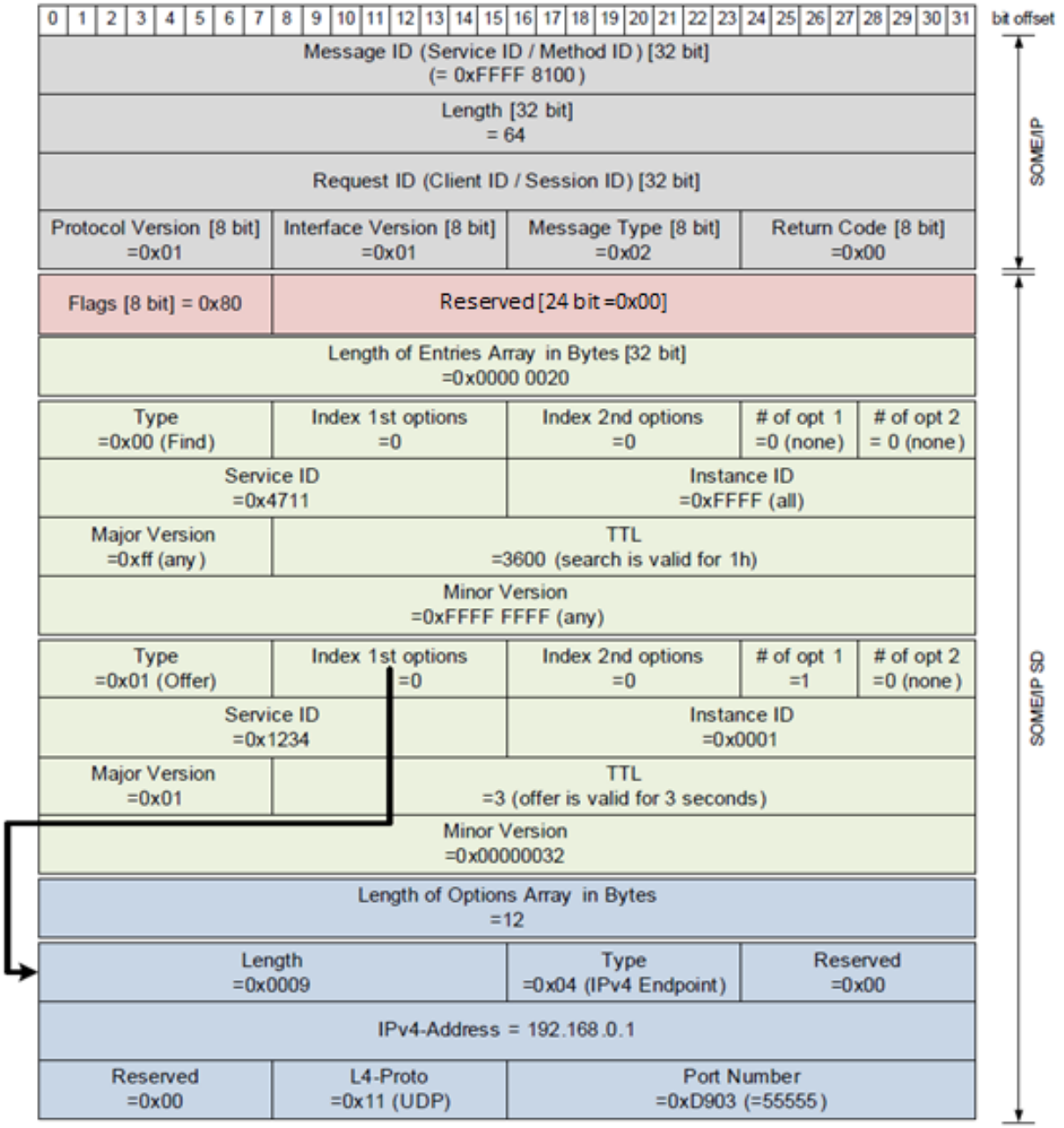

Figure 4.2: SOME/IP-SD Example PDU

4.1.2.2 SOME/IP-SD Header

[PRS_SOMEIPSD_00252] ⌈SOME/IP-SD shall be transported using SOME/IP.⌋ *(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00252] can be seen in Figure 4.2.

[PRS_SOMEIPSD_00253] ⌈The SOME/IP-SD Header shall start with an 8 Bit field called `flags`.⌋*(RS_SOMEIPSD_00006)*

See a representation of Flags in Figure 4.3.

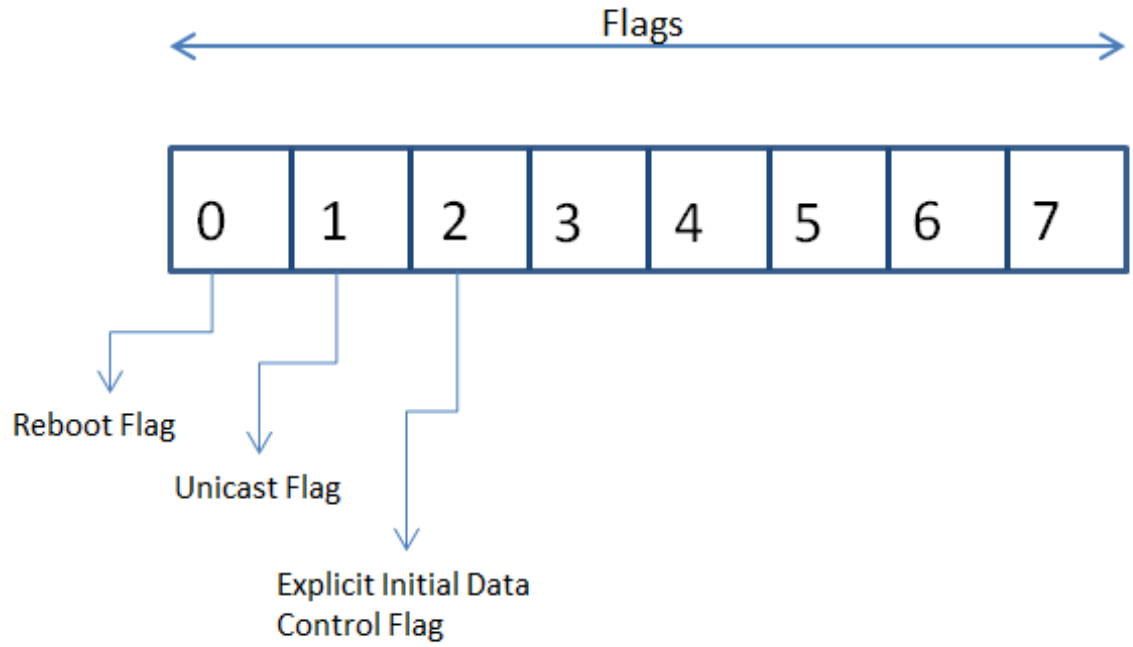

Figure 4.3: Flags in SOME/IP-SD

[PRS_SOMEIPSD_00254] ⌈The first flag of the SOME/IP-SD Flags field (highest order bit) shall be called `Reboot Flag`.⌋*(RS_SOMEIPSD_00006)*

For flags see Figure 4.3.

[PRS_SOMEIPSD_00255] ⌈The Reboot Flag of the SOME/IP-SD Header shall be set to one for all messages after reboot until the Session-ID in the SOME/IP-Header wraps around and thus starts with 1 again. After this wrap around the Reboot Flag is set to 0.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00256] ⌈The information for the reboot flag and the Session ID shall be kept for multicast and unicast separately.⌋*(RS_SOMEIPSD_00002, RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00631] ⌈The information for the reboot flag and the Session ID shall be kept for every sender-receiver relation (i.e. source address and destination address) separately.⌋*(RS_SOMEIPSD_00002, RS_SOMEIPSD_00003)*

Note:
This means there shall be separate counters for sending and receiving.

Sending

- There shall be a counter for multicast.
- There shall be a separate counter for each peer for unicast.

Receiving

- There shall be a counter for each peer for multicast.
- There shall be a counter for each peer for unicast.

[PRS_SOMEIPSD_00258] ⌈The detection of a reboot shall be done as follows (with the new values of the current packet from the communication partner and old the last value received before):

if old.reboot==0 and new.reboot==1 then Reboot detected
OR
if old.reboot==1 and new.reboot==1 and old.session_id>=new.session_id then Reboot detected
⌋*(RS_SOMEIPSD_00006)*

Note:
The following is not enough since we do not have reliable communication:
if new.reboot==1 and old.session_id>=new.session_id then Reboot detected

[PRS_SOMEIPSD_00259] ⌈The second flag of the SOME/IP-SD Flags (second highest order bit) shall be called Unicast Flag.⌋*(RS_SOMEIPSD_00002)*

For flags see Figure 4.3.

[PRS_SOMEIPSD_00540] ⌈The Unicast Flag of the SOME/IP-SD Header shall be set to Unicast (that means 1) for all SD Messages since this means that receiving using unicast is supported.⌋*(RS_SOMEIPSD_00002)*

Note:
The Unicast Flag is left over from historical SOME/IP versions and is only kept for compatibility reasons. Its use besides this is very limited.

[PRS_SOMEIPSD_00700] ⌈The third flag of the SOME/IP-SD Flags (third highest order bit) shall be called Explicit Initial Data Control Flag and shall mean that the ECU supports explicit initial data control.⌋*(RS_SOMEIPSD_00002)*

For flags see Figure 4.3.

Note:
This means that the ECU understands and honors the Initial Data Requested Flag inside of Eventgroup Entries.

[PRS_SOMEIPSD_00701] ⌈The Explicit Initial Data Control Flag shall always be set to 1, meaning the ECU supports this feature.⌋*(RS_SOMEIPSD_00002)*

Note:
This flag allows compatibility to older SOME/IP-SD implementations (e.g. AUTOSAR

4.2), which do not support this feature and have set this flag to 0. New versions shall all support the feature, so they have to set this flag always to 1.

[PRS_SOMEIPSD_00702] ⌈Undefined bits within the Flag field shall be set to '0' when sending and ignored on receiving.⌋ *(RS_SOMEIPSD_00002)*

[PRS_SOMEIPSD_00261] ⌈After the Flags the SOME/IP-SD Header shall have a field of 24 bits called Reserved.⌋ *(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00262] ⌈After the SOME/IP-SD Header the Entries Array shall follow.⌋ *(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00263] ⌈The entries shall be processed exactly in the order they arrive.⌋ *(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00264] ⌈After the Entries Array in the SOME/IP-SD Header an Option Array shall follow.⌋ *(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00265] ⌈The Entries Array and the Options Array of the SOME/IP-SD message shall start with a `length` field as uint32 that counts the number of bytes of the following data; i.e. the entries or the options.⌋ *(RS_SOMEIPSD_00006)*

4.1.2.3 Entry Format

[PRS_SOMEIPSD_00266] ⌈The service discovery shall support multiple entries that are combined in one service discovery message.⌋ *(RS_SOMEIPSD_00006)*

Note:
The entries are used to synchronize the state of services instances and the Publish/-Subscribe handling.

[PRS_SOMEIPSD_00267] ⌈Two types of entries exist: A Service Entry Type for Services and an Eventgroup Entry Type for Eventgroups.⌋ *(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00268] ⌈A Service Entry Type shall be 16 Bytes of size and include the following fields in this order:

- Type Field [uint8]: encodes FindService (0x00) and OfferService (0x01).
- Index First Option Run [uint8]: Index of this runs first option in the option array.
- Index Second Option Run [uint8]: Index of this runs second option in the option array.
- Number of Options 1 [uint4]: Describes the number of options the first option run uses.
- Number of Options 2 [uint4]: Describes the number of options the second option run uses.
- Service-ID [uint16]: Describes the Service ID of the Service or Service-Instance this entry is concerned with.

- Instance ID [uint16]: Describes the Service Instance ID of the Service Instance this entry is concerned with or is set to 0xFFFF if all service instances of a service are meant.
- Major Version [uint8]: Encodes the major version of the service (instance).
- TTL [uint24]: Describes the lifetime of the entry in seconds.
- Minor Version [uint32]: Encodes the minor version of the service.

⌋(*RS_SOMEIPSD_00006*)

[PRS_SOMEIPSD_00268] is shown in Figure 4.1.

[PRS_SOMEIPSD_00269] ⌈SOME/IP-SD Service Entry Type shall follow this format:

- Type Field [uint8]
- Index 1st Options [uint8]
- Index 2nd Options [uint8]
- Number of Options 1 [uint4]
- Number of Options 2 [uint4]
- Service-ID [uint16]
- Instance ID [uint16]
- Major Version [uint8]
- TTL [uint24]
- Minor Version [uint32]

⌋(*RS_SOMEIPSD_00006*)

The format of [PRS_SOMEIPSD_00269] is shown in Figure 4.4.

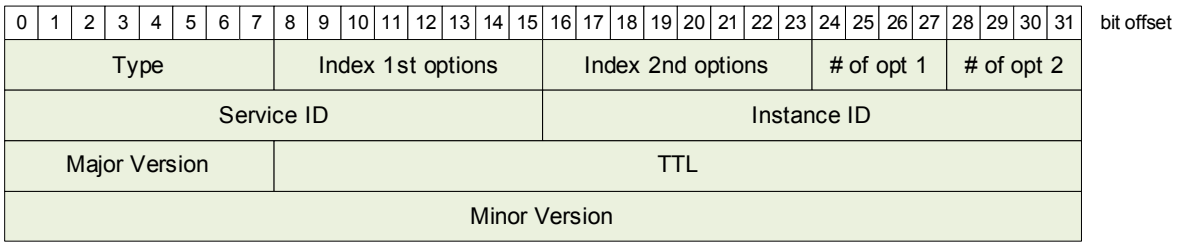

Figure 4.4: SOME/IP-SD Service Entry Type

[PRS_SOMEIPSD_00270] ⌈An Eventgroup Entry shall be 16 Bytes of size and include the following fields in this order:

- Type Field [uint8]: encodes Subscribe (0x06), and SubscribeAck (0x07).
- Index First Option Run [uint8]: Index of this runs first option in the option array.

- Index Second Option Run [uint8]: Index of this runs second option in the option array.
- Number of Options 1 [uint4]: Describes the number of options the first option run uses.
- Number of Options 2 [uint4]: Describes the number of options the second option run uses.
- Service-ID [uint16]: Describes the Service ID of the Service or Service Instance this entry is concerned with.
- Instance ID [uint16]: Describes the Service Instance ID of the Service Instance this entry is concerned with or is set to 0xFFFF if all service instances of a service are meant.
- Major Version [uint8]: Encodes the major version of the service instance this eventgroup is part of.
- TTL [uint24]: Descibes the lifetime of the entry in seconds.
- Reserved [uint8]: Shall be set to 0x00.
- Initial Data Requested Flag [1 bit] (I Flag): Shall be set to 1, if initial data shall be sent by Server (see [PRS_SOMEIPSD_00703]).
- Reserved2 [uint3]: Shall be set to 0x0, if not specified otherwise (see [PRS_SOMEIPSD_00391] and [PRS_SOMEIPSD_00394]).
- Counter [uint4]: Is used to differentiate identical Subscribe Eventgroups of the same subscriber. Set to 0x0 if not used.
- Eventgroup ID [uint16]: Transports the ID of an Eventgroup.

⌋(*RS_SOMEIPSD_00006*)

SOME/IP-SD Eventgroup Entry Type is shown in Figure 4.5.

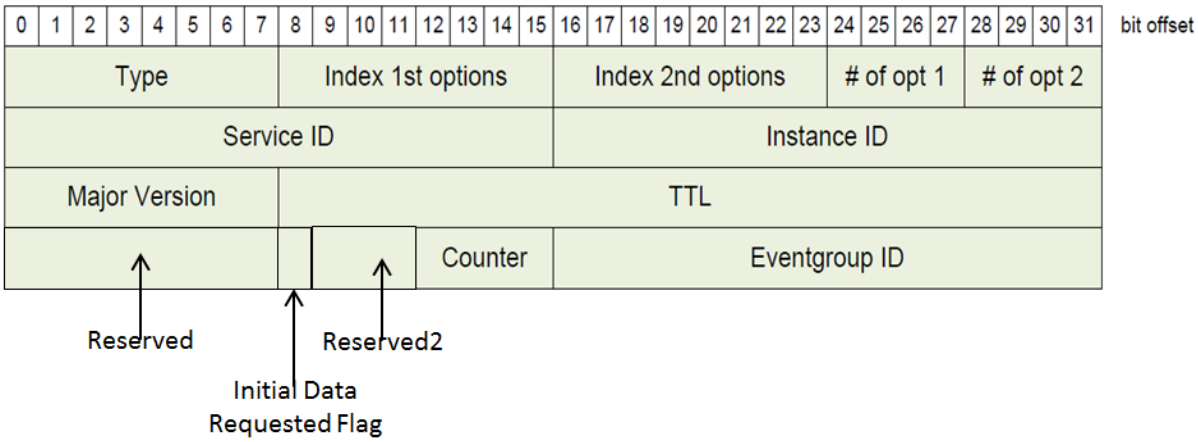

Figure 4.5: SOME/IP-SD Eventgroup Entry Type

Referencing Options from Entries

Using the following fields of the entries, options are referenced by the entries:

- Index First Option Run: Index into array of options for first option run. Index 0 means first of SOME/IP-SD packet.
- Index Second Option Run: Index into array of options for second option run. Index 0 means first of SOME/IP-SD packet.
- Number of Options 1: Length of first option run. Length 0 means no option in option run.
- Number of Options 2: Length of second option run. Length 0 means no option in option run.

Two different option runs exist: First Option Run and Second Option Run.

Rationale for the support of two option runs: Two different types of options are expected: options common between multiple SOME/IP-SD entries and options different for each SOME/IP-SD entry. Supporting two different options runs is the most efficient way to support these two types of options, while keeping the wire format highly efficient.

[PRS_SOMEIPSD_00341] ⌈Each option run shall reference the first option and the number of options for this run.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00342] ⌈If the number of options is set to zero, the option run is considered empty.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00343] ⌈For empty runs the Index (i.e. Index First Option Run and/or Index Second Option Run) shall be set to zero.⌋*(RS_SOMEIPSD_00025)*

4.1.2.4 Options Format

Options are used to transport additional information to the entries. This includes for instance the information how a service instance is reachable (IP-Address, Transport Protocol, Port Number).

[PRS_SOMEIPSD_00273] ⌈In order to identify the option type every option shall start with:

- Length [uint16]: Specifies the length of the option in Bytes.
- Type [uint8]: Specifying the type of the option.
- Discardable Flag [1 bit]: Specifies if the option can be discarded.
- Bit 1 to bit 7 are reserved and shall be 0.

⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00274] ⌈The length field shall cover all bytes of the option except the length field and type field.⌋*(RS_SOMEIPSD_00006)*

If another option follows an option with variable length, the 2nd option may be misaligned depending on the length of the 1st option.

[PRS_SOMEIPSD_00275] ⌈The discardable flag shall be set to 1 if the option can be discarded by a receiving ECU that does not support this option.⌋*(RS_SOMEIPSD_00006)*

Configuration Option

The configuration option is used to transport arbitrary configuration strings. This allows to encode additional information like the name of a service or its configuration.
[PRS_SOMEIPSD_00276] ⌈The format of the Configuration Option shall be as follows:

- Length [uint16]: Shall be set to the total number of bytes occupied by the configuration option, excluding the 16 bit length field and the 8 bit type flag.
- Type [uint8]: Shall be set to 0x01.
- Discardable Flag [1 bit]: Shall be set to 1 if the Option can be discarded by the receiver.
- Bit 1 to bit 7 are reserved and shall be 0.
- ConfigurationString [dyn length]: Shall carry the configuration string.

⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00277] ⌈The Configuration Option shall specify a set of name-value-pairs based on the DNS TXT and DNS-SD format.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00009)*

[PRS_SOMEIPSD_00278] ⌈The format of the configuration string shall start with a single byte length field that describes the number of bytes following this length field. After the length field a character sequence with the specified length shall follow.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00279] ⌈After each character sequence another length field and a following character sequence are expected until a length field shall be set to 0x00.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00280] ⌈After a length field is set to 0x00 no characters shall follow.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00281] ⌈A character sequence shall encode a key and optionally a value.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00282] ⌈The character sequences shall contain an equal character ("=", 0x3D) to divide key and value.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00283] ⌈The key shall not include an equal character and shall be at least one non-whitespace character. The characters of "Key" shall be printable US-ASCII values (0x20-0x7E), excluding '=' (0x3D).⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00284] ⌈The "=" shall not be the first character of the sequence.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00285] ⌈For a character sequence without an '=' that key shall be interpreted as present.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00286] ⌈For a character sequence ending on an '=' that key shall be interpreted as present with empty value.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00287] ⌈Multiple entries with the same key in a single Configuration Option shall be supported.⌋*(RS_SOMEIPSD_00006)*

[PRS_SOMEIPSD_00289] ⌈Format of the Configuration Option shall be:

- Length [uint16]
- Type [uint8]: `0x01`
- Reserved [uint8]: `0x00`
- Zero-terminated Configuration String [variable size]

⌋*(RS_SOMEIPSD_00006)*

The format of the Configuration Option is shown in Figure 4.6.

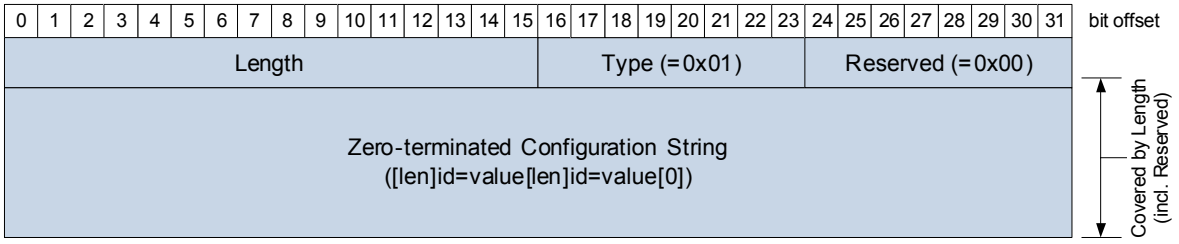

Figure 4.6: SOME/IP-SD Configuration Option

Example for SOME/IP-SD Configuration Option

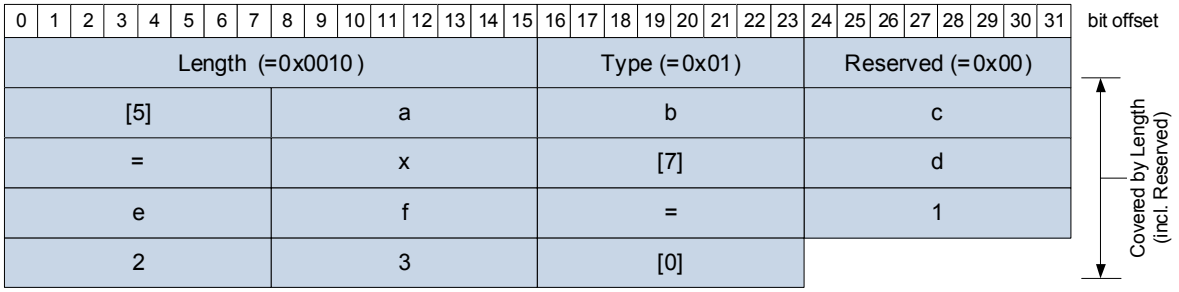

Figure 4.7: SOME/IP-SD Configuration Option Example

Load Balancing Option

The Load Balancing option is used to prioritize different instances of a service, so that a client chooses the service instance based on these settings. This option will be attached to Offer Service entries.

[PRS_SOMEIPSD_00542] ⌈The Load Balancing Option shall carry a Priority and Weight like the DNS-SRV records, which shall be used for load balancing different service instances.⌋*(RS_SOMEIPSD_00011)*

[PRS_SOMEIPSD_00711] ⌈When looking for all service instances of a service (Service Instance set to 0xFFFF), the client shall choose the service instance with highest priority that also matches client specific criteria.⌋*(RS_SOMEIPSD_00011)*

Note: Client specific criteria may be applied by the client application when choosing one of the offered service instances. They are not defined in this specification, and could e.g. restrict the range of appropriate instance IDs.

[PRS_SOMEIPSD_00712] ⌈When having more than one service instance with highest priority (lowest value in Priority field) the service instance shall be chosen randomly based on the weights of the service instances. The probability of choosing a service instance shall be the weight of a service instance divided by the sum of the weights of all considered service instances.⌋*(RS_SOMEIPSD_00011)*

[PRS_SOMEIPSD_00713] ⌈If an Offer Service entry references no Load Balancing option and several service instances are offered, the client shall handle the service instances without Load Balancing option as though they had the lowest priority.⌋*(RS_SOMEIPSD_00011)*

[PRS_SOMEIPSD_00714] ⌈When looking for a specific service instances of a service (Service Instance set to any value other than 0xFFFF), the evaluation of the Load Balancing Option does not apply.⌋*(RS_SOMEIPSD_00011)*

[PRS_SOMEIPSD_00544] ⌈The Format of the Load Balancing Option shall be as follows:

- Length [uint16]: Shall be set to 0x0005.
- Type [uint8]: Shall be set to 0x02.
- Discardable Flag [1 bit]: Shall be set to 1 if the Option can be discarded by the receiver.
- Bit 1 to bit 7 are reserved and shall be 0.
- Priority [uint16]: Carries the Priority of this instance. Lower value means higher priority.
- Weight [uint16]: Carries the Weight of this instance. Large value means higher probability to be chosen.

⌋*(RS_SOMEIPSD_00011)*

The format of the Load Balancing Option is shown in Figure 4.8.

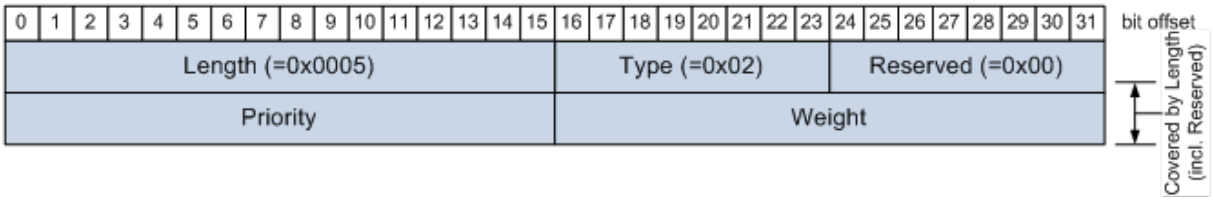

Figure 4.8: SOME/IP-SD Load Balancing Option

IPv4 Endpoint Option

The IPv4 Endpoint Option is used by a SOME/IP-SD instance to signal the relevant endpoint(s). Endpoints include the local IP address, the transport layer protocol (e.g. UDP or TCP), and the port number of the sender. These ports are used for the events and notification events as well.

[PRS_SOMEIPSD_00305] ⌈The IPv4 Endpoint Option shall use the Type 0x04.⌋ *(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00306] ⌈The IPv4 Endpoint Option shall specify the IPv4-Address, the transport layer protocol (ISO/OSI layer 4) used, and its Port Number.⌋ *(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00307] ⌈The Format of the IPv4 Endpoint Option shall be as follows:

- Length [uint16]: Shall be set to 0x0009.
- Type [uint8]: Shall be set to 0x04.
- Discardable Flag [1 bit]: Shall be set to 0.
- Bit 1 to bit 7 are reserved and shall be 0.
- IPv4-Address [uint32]: Shall transport the unicast IP-Address as four Bytes.
- Reserved [uint8]: Shall be set to 0x00.
- Transport Protocol (L4-Proto) [uint8]: Shall be set to the transport layer protocol (ISO/OSI layer 4) based on the IANA/IETF types (0x06: TCP, 0x11: UDP).
- Transport Protocol Port Number (L4-Port) [uint16]: Shall be set to the port of the layer 4 protocol.

⌋ *(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

SOME/IP-SD IPv4 Endpoint Option is shown in Figure 4.9.

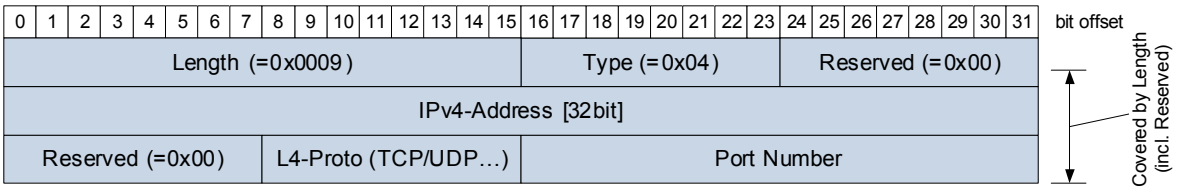

Figure 4.9: SOME/IP-SD IPv4 Endpoint Option

[PRS_SOMEIPSD_00310] ⌈The server shall use the IPv4 Endpoint Option with Offer Service entries to signal the endpoints it serves the service on. That is upto one UDP endpoint and upto one TCP endpoint.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00380] ⌈The endpoints the server referenced with an Offer Service entry shall also be used as source of events. That is source IP address and source port numbers for the transport protocols in the endpoint option. The client shall use the IPv4 Endpoint Options with Subscribe Eventgroup entries to signal its IP address and its UDP and/or TCP port numbers, on which it is ready to receive the events.⌋ *(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00807] ⌈The client shall use the IPv4 Endpoint Option with Subscribe Eventgroup entries to signal the IP address and the UDP and/or TCP port numbers, on which it is ready to receive the events.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

IPv6 Endpoint Option

The IPv6 Endpoint Option is used by a SOME/IP-SD instance to signal the relevant endpoint(s). Endpoints include the local IP address, the transport layer protocol (e.g. UDP or TCP), and the port number of the sender. These ports are used for the events and notification events as well.

[PRS_SOMEIPSD_00314] ⌈The IPv6 Endpoint Option shall use the Type 0x06.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00315] ⌈The Format of the IPv6 Endpoint Option shall be as follows:

- Length [uint16]: Shall be set to 0x0015.
- Type [uint8]: Shall be set to 0x06.
- Discardable Flag [1 bit]: Shall be set to 0.
- Bit 1 to bit 7 are reserved and shall be 0.
- IPv6-Address [uint128]: Shall transport the unicast IP-Address as 16 Bytes.
- Reserved [uint8]: Shall be set to 0x00.

- Transport Protocol (L4-Proto) [uint8]: Shall be set to the transport layer protocol (ISO/OSI layer 4) based on the IANA/IETF types (0x06: TCP, 0x11: UDP).
- Transport Protocol Port Number (L4-Port) [uint16]: Shall be set to the transport layer port(e.g. 30490).

⌋(*RS_SOMEIPSD_00006*, *RS_SOMEIPSD_00010*)

SOME/IP-SD IPv6 Endpoint Option shall be as shown in Figure 4.10

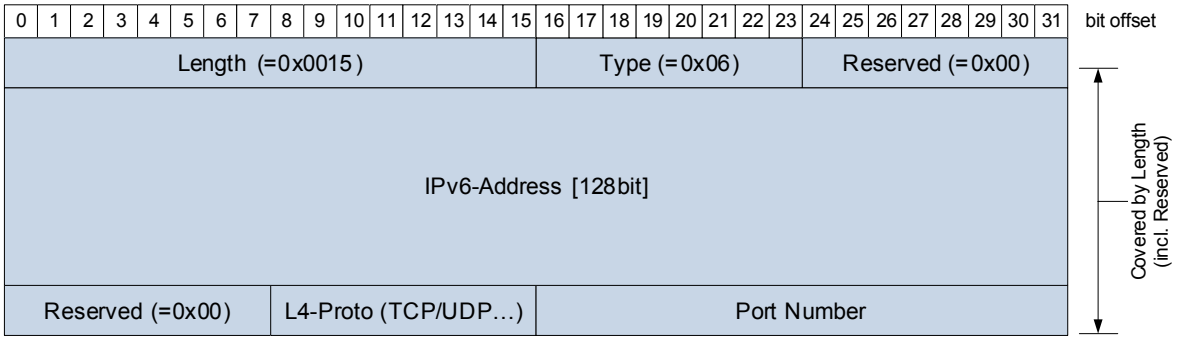

Figure 4.10: SOME/IP-SD IPv6 Endpoint Option

[PRS_SOMEIPSD_00319] ⌈The server shall use the IPv6 Endpoint Option with Offer Service entries to signal the endpoints the services is available on. That is upto one UDP endpoint and upto one TCP endpoint.⌋(*RS_SOMEIPSD_00006*, *RS_SOMEIPSD_00010*)

[PRS_SOMEIPSD_00320] ⌈The endpoints the server referenced with an Offer Service entry shall also be used as source of events. That is source IP address and source port numbers for the transport protocols in the endpoint option.⌋(*RS_SOMEIPSD_00006*, *RS_SOMEIPSD_00010*)

[PRS_SOMEIPSD_00321] ⌈The client shall use the IPv6 Endpoint Option with Subscribe Eventgroup entries to signal the IP address and the UDP and/or TCP port numbers, on which it is ready to receive the events.⌋(*RS_SOMEIPSD_00006*, *RS_SOMEIPSD_00010*)

IPv4 Multicast Option

The IPv4 Multicast Option is used by the server to announce the IPv4 multicast address, the transport layer protocol (ISO/OSI layer 4), and the port number the multicast events and multicast notification events are sent to. As transport layer protocol currently only UDP is supported.

[PRS_SOMEIPSD_00323] ⌈The IPv4 Multicast Option shall be referenced by Subscribe Eventgroup Ack entries.⌋(*RS_SOMEIPSD_00003*)

[PRS_SOMEIPSD_00324] ⌈The IPv4 Multicast Option shall use the Type 0x14.⌋*(RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00325] ⌈The IPv4 Multicast Option shall specify the IPv4-Address, the transport layer protocol (ISO/OSI layer 4) used, and its Port Number.⌋*(RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00326] ⌈The Format of the IPv4 Endpoint Option shall be as follows:

- Length [uint16]: Shall be set to 0x0009.
- Type [uint8]: Shall be set to 0x14.
- Discardable Flag [1 bit]: Shall be set to 0.
- Bit 1 to bit 7 are reserved and shall be 0.
- IPv4-Address [uint32]: Shall transport the multicast IP-Address as four Bytes.
- Reserved [uint8]: Shall be set to 0x00.
- Transport Protocol (L4-Proto) [uint8]: Shall be set to the transport layer protocol (ISO/OSI layer 4) based on the IANA/IETF types (0x11: UDP).
- Transport Protocol Port Number (L4-Port) [uint16]: Shall be set to the port of the layer 4 protocol.

⌋*(RS_SOMEIPSD_00003)*

SOME/IP-SD IPv4 Multicast Option shall be as shown in Figure 4.11

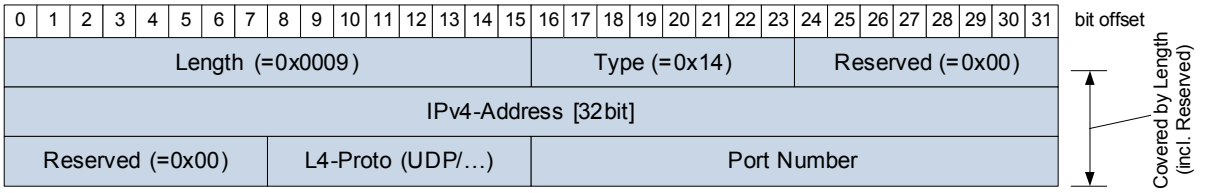

Figure 4.11: SOME/IP-SD IPv4 Multicast Option

[PRS_SOMEIPSD_00329] ⌈The server shall reference the IPv4 Multicast Option, which encodes the IPv4 Multicast Address and Port Number the server will send multicast events and notification events to.⌋*(RS_SOMEIPSD_00003)*

IPv6 Multicast Option

The IPv6 Multicast Option is used by the server to announce the IPv6 multicast address, the layer 4 protocol, and the port number the multicast events and multicast notifications events are sent to. For the transport layer protocol (ISO/OSI layer 4) currently only UDP is supported.

[PRS_SOMEIPSD_00331] ⌈The IPv6 Multicast Option shall use the Type 0x16.⌋*(RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00332] ⌈The IPv6 Multicast Option shall specify the IPv6-Address, the transport layer protocol (ISO/OSI layer 4) used, and its Port Number.⌋*(RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00333] ⌈The Format of the IPv6 Multicast Option shall be as follows:

- Length [uint16]: Shall be set to 0x0015.
- Type [uint8]: Shall be set to 0x16.
- Discardable Flag [1 bit]: Shall be set to 0.
- Bit 1 to bit 7 are reserved and shall be 0.
- IPv6-Address [uint128]: Shall transport the multicast IP-Address as 16 Bytes.
- Reserved [uint8]: Shall be set to 0x00.
- Transport Protocol (L4-Proto) [uint8]: Shall be set to the transport layer protocol (ISO/OSI layer 4) based on the IANA/IETF types (0x11: UDP).
- Transport Protocol Port Number (L4-Port) [uint16]: Shall be set to the port of the layer 4 protocol.

⌋*(RS_SOMEIPSD_00003)*

SOME/IP-SD IPv6 Multicast Option shall be as shown in Figure 4.12.

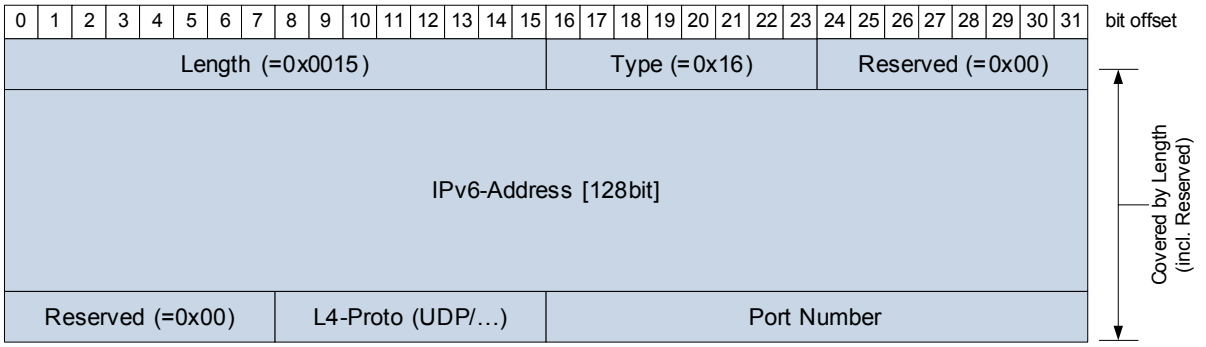

Figure 4.12: SOME/IP-SD IPv6 Multicast Option

[PRS_SOMEIPSD_00545] ⌈The IPv6 Multicast Option and not the IPv6 Endpoint Option shall be referenced by Subscribe Eventgroup Ack messages.⌋*(RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00336] ⌈The server shall reference the IPv6 Multicast Option, which encodes the IPv6 Multicast Address and Port Number the server will send multicast events and notification events to.⌋*(RS_SOMEIPSD_00003)*

IPv4 SD Endpoint Option

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

Note:
This is used to identify the SOME/IP-SD Instance even in cases in which the IP-Address and/or Port Number cannot be used. A use case would be a proxy service discovery on one ECU which handles the multicast traffic for another ECU.

SOME/IP-SD IPv4 SD Endpoint Option is shown in Figure 4.13

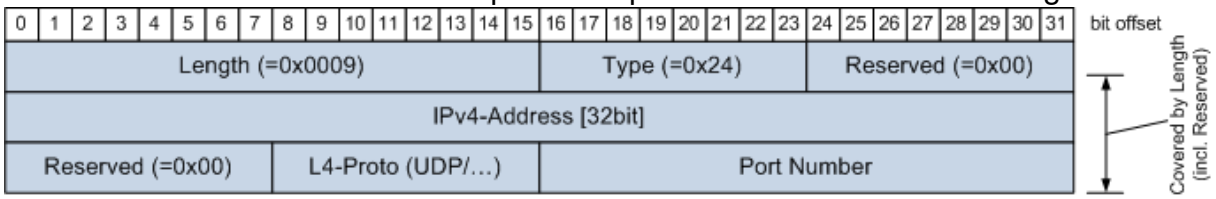

Figure 4.13: SOME/IP-SD IPv4 SD Endpoint Option

[PRS_SOMEIPSD_00547] ⌈The IPv4 SD Endpoint Option shall be included in any SD message up to 1 time.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00650] ⌈The IPv4 SD Endpoint Option shall only be included if the SOME/IP-SD message is transported over IPv4.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00651] ⌈The IPv4 SD Endpoint Option shall be the first option in the options array, if existing.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00548] ⌈The IPv4 SD Endpoint Option shall not be referenced by any SD Entry.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00549] ⌈If the IPv4 SD Endpoint Option is included in the SD message, the receiving SD implementation shall use the content of this option instead of the Source IP Address and Source Port.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

Note:
This is important for answering the received SD message (e.g. Offer after Find or Subscribe after Offer or Subscribe Ack after Subscribe) as well as the reboot detection (channel based on SD Endpoint Option and not out addresses).

[PRS_SOMEIPSD_00550] ⌈The IPv4 SD Endpoint Option shall use the Type 0x24.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00551] ⌈The IPv4 SD Endpoint Option shall specify the IPv4-Address, the transport layer protocol (ISO/OSI layer 4) and Port Number of the sender used for Service Discovery.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00552] ⌈The Format of the IPv4 SD Endpoint Option shall be as follows:

- Length [uint16]: Shall be set to 0x0015.
- Type [uint8]: Shall be set to 0x24.
- Discardable Flag [1 bit]: Shall be set to 0.
- Bit 1 to bit 7 are reserved and shall be 0.
- IPv4-Address [uint32]: Shall transport the unicast IP-Address of SOME/IP-SD as four Bytes.
- Reserved [uint8]: Shall be set to 0x00.
- Transport Protocol (L4-Proto) [uint8]: Shall be set to the transport layer protocol of SOME/IP-SD (currently: 0x11 UDP).
- Transport Protocol Port Number (L4-Port) [uint16]: Shall be set to the transport layer port of SOME/IP-SD (currently: 30490).

⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

IPv6 SD Endpoint Option

The Ipv6 SD Endpoint Option is used to transport the endpoint (i.e. IP-Address and Port) of the senders SD implementation. This is used to identify the SOME/IP-SD Instance even in cases in which the IP-Address and/or Port Number cannot be used. SOME/IP-SD IPv6 SD Endpoint Option is shown in Figure 4.14

Note:
A use case would be a proxy service discovery on one ECU which handles the multicast traffic for another ECU.

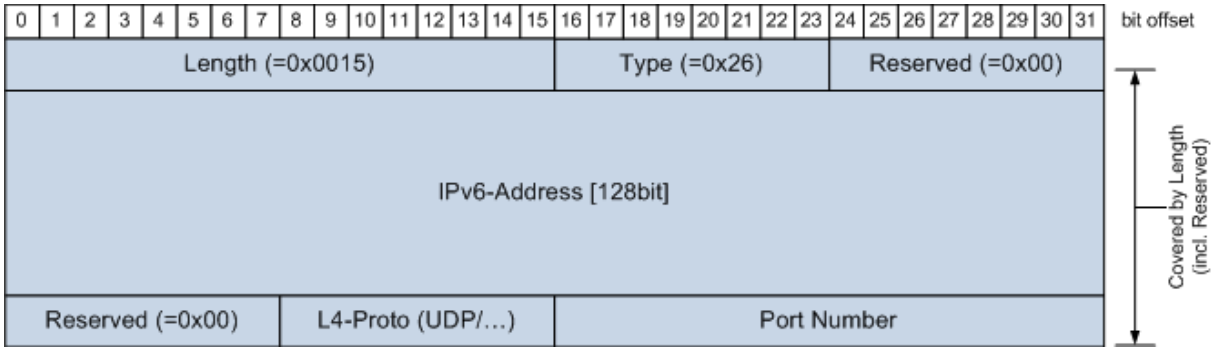

Figure 4.14: SOME/IP-SD IPv6 SD Endpoint Option

[PRS_SOMEIPSD_00554] ⌈The IPv6 SD Endpoint Option may be included in any SD message up to 1 time.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00654] ⌈The IPv6 SD Endpoint Option shall be the first option in the options array, if existing.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00555] ⌈The IPv6 SD Endpoint Option shall not be referenced by any SD Entry.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00556] ⌈If the IPv6 SD Endpoint Option is included in the SD message, the receiving SD implementation shall use the content of this option instead of the Source IP Address and Source Port for answering this SD messages.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00557] ⌈The IPv6 SD Endpoint Option shall use the Type 0x26.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00558] ⌈The IPv6 SD Endpoint Option shall specify the IPv6-Address, the transport layer protocol (ISO/OSI layer 4) and Port Number of the sender used for Service Discovery.⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

[PRS_SOMEIPSD_00559] ⌈The Format of the IPv6 SD Endpoint Option shall be as follows:

- Length [uint16]: Shall be set to 0x0015.
- Type [uint8]: Shall be set to 0x26.
- Discardable Flag [1 bit]: Shall be set to 0.
- Bit 1 to bit 7 are reserved and shall be 0.
- IPv6-Address [uint128]: Shall transport the unicast IP-Address of SOME/IP-SD as 16 Bytes.
- Reserved [uint8]: Shall be set to 0x00.
- Transport Protocol (L4-Proto) [uint8]: Shall be set to the transport layer protocol of SOME/IP-SD (currently: 0x11 UDP).
- Transport Protocol Port Number (L4-Port) [uint16]: Shall be set to the transport layer port of SOME/IP-SD (currently: 30490).

⌋*(RS_SOMEIPSD_00006, RS_SOMEIPSD_00010)*

4.1.2.5 Service Entries

Find Service Entry

[PRS_SOMEIPSD_00350] ⌈The Find Service entry type shall be used for finding service instances and shall only be sent if the current state of a service is unknown (no current Service Offer was received and is still valid).⌋*(RS_SOMEIPSD_00008, RS_SOMEIPSD_00021)*

[PRS_SOMEIPSD_00351] ⌈Find Service entries shall set the entry fields in the following way:

- Type shall be set to 0x00 (FindService).
- Service ID shall be set to the Service ID of the service that shall be found.
- Instance ID shall be set to 0xFFFF, if all service instances shall be returned. It shall be set to the Instance ID of a specific service instance, if just a single service instance shall be returned.
- Major Version shall be set to 0xFF, that means that services with any version shall be returned. If set to value different than 0xFF, services with this specific major version shall be returned only.
- Minor Version shall be set to 0xFFFF FFFF, that means that services with any version shall be returned. If set to a value different to 0xFFFF FFFF, services with this specific minor version shall be returned only.
- TTL shall be set to the lifetime of the Find Service entry. After this lifetime the Find Service entry shall be considered not existing.
- If TTL is set to 0xFFFFFF, the Find Service entry shall be considered valid until the next reboot.
- TTL shall not be set to 0x000000 since this is considered to be the Stop Offer Service Entry.

⌋*(RS_SOMEIPSD_00008, RS_SOMEIPSD_00021)*

[PRS_SOMEIPSD_00528] ⌈A sender shall not reference Endpoint Options nor Multicast Options in a Find Service Entry.⌋*(RS_SOMEIPSD_00008, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00529] ⌈A receiver shall ignore Endpoint Options and Multicast Options in a Find Service Entry.⌋*(RS_SOMEIPSD_00013, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00530] ⌈Other Options (neither Endpoint nor Multicast Options), shall still be allowed to be used in a Find Service Entry.⌋*(RS_SOMEIPSD_00013, RS_SOMEIPSD_00025)*

Offer Service Entry

[PRS_SOMEIPSD_00355] ⌈The Offer Service entry type shall be used to offer a service to other communication partners.⌋*(RS_SOMEIPSD_00013)*

[PRS_SOMEIPSD_00356] ⌈Offer Service entries shall set the entry fields in the following way:

- Type shall be set to 0x01 (OfferService).
- Service ID shall be set to the Service ID of the service instance offered.

- Instance ID shall be set to the Instance ID of the service instance that is offered.
- Major Version shall be set to the Major Version of the service instance that is offered.
- Minor Version shall be set to the Minor Version of the service instance that is offered.
- TTL shall be set to the lifetime of the service instance. After this lifetime the service instance shall considered not been offered.
- If TTL is set to 0xFFFFFF, the Offer Service entry shall be considered valid until the next reboot.
- TTL shall not be set to 0x000000 since this is considered to be the Stop Offer Service Entry.

⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00357] ⌈Offer Service entries shall always reference at least an IPv4 or IPv6 Endpoint Option to signal how the service is reachable.⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00358] ⌈For each Transport Layer Protocol needed for the service (i.e. UDP and/or TCP) an IPv4 Endpoint option shall be added if IPv4 is supported.⌋ (*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00359] ⌈For each Transport Layer Protocol needed for the service (i.e. UDP and/or TCP) an IPv6 Endpoint option shall be added if IPv6 is supported.⌋ (*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

Stop Offer Service Entry

[PRS_SOMEIPSD_00363] ⌈The Stop Offer Service entry type shall be used to stop offering service instances.⌋(*RS_SOMEIPSD_00014*)

[PRS_SOMEIPSD_00364] ⌈Stop Offer Service entries shall set the entry fields exactly like the Offer Service entry they are stopping, except:

- TTL shall be set to 0x000000.

⌋(*RS_SOMEIPSD_00014*)

Usage of Options in Entries

[PRS_SOMEIPSD_00583] ⌈

Table 4.1 shall show the SOME/IP-SD Option types in relation to Entry types which are allowed:

⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00008*, *RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00014*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00016*)

	Endpoint	Multicast	Configuration	Load Balancing
FindService	0	0	0-1	0-1
OfferService	1-2	0	0-1	0-1
StopOffer Service	1-2	0	0-1	0-1
Subscribe Eventgroup	1-2	0	0-1	0-1
StopSubscribe Eventgroup	1-2	0	0-1	0-1
Subscribe EventgroupAck	0	0-1	0-1	0-1
Subscribe EventgroupNack	0	0	0-1	0-1

Table 4.1: Allowed Option Types for Entry Types

4.1.2.6 Endpoint Handling for Services and Events

[PRS_SOMEIPSD_00476] ⌈The Service Discovery shall overwrite IP Addresses and Port Numbers with those transported in Endpoint and Multicast Options if the statically configured values are different from those in these options.⌋(*RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00360] ⌈The IP addresses and port numbers of the Endpoint Options shall also be used for transporting events and notification events.⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00361] ⌈In case of UDP the endpoint option shall be used for the source address and the source port of the events and notification events, it is also the address the client can send method requests to.⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00362] ⌈In case of TCP the endpoint option shall be used for the IP address and port the client needs to open a TCP connection in order to receive events using TCP.⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00801] ⌈SOME/IP shall allow services to use UDP and TCP at the same time.⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

[PRS_SOMEIPSD_00802] ⌈Which message is sent by which underlying transport protocol shall be determined by configuration: A Service can use UDP and TCP endpoints at the same time. But per element of the service it shall to be configured whether TCP or UDP is used.⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00025*)

Note: It needs to be restricted in the configuration which methods and which events are provided over TCP/UDP. This also means that the same event can not be provided over TCP and UDP.

Service Endpoints

The referenced Endpoint Options of the Offer Service entries denotes the

- IP Address and Port Numbers the service instance is reachable at the server.
- IP Address and Port Numbers the service instance sends the events from.

[PRS_SOMEIPSD_00480] ⌈Events of this service instance shall not be sent from any other Endpoints than those given in the Endpoint Options of the Offer Service entries.⌋ *(RS_SOMEIPSD_00013, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00481] ⌈If an ECU offers multiple service instances, SOME/IP messages of these service instances shall be differentiated by the information transported in the Endpoint Options referenced by the Offer Service entries.⌋ *(RS_SOMEIPSD_00013, RS_SOMEIPSD_00025)*

Eventgroup Endpoints

[PRS_SOMEIPSD_00484] ⌈The Endpoint Options referenced in the Subscribe Eventgroup entries shall also be used to send unicast UDP or TCP SOME/IP events for this Service Instance.⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00025)*

Thus the Endpoint Options referenced in the Subscribe Eventgroup entries are the IP Address and the Port Numbers on the client side.

[PRS_SOMEIPSD_00486] ⌈TCP events are transported using the TCP connection the client has opened to the server before sending the Subscribe Eventgroup entry. See Chapter 4.1.2.4.⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00487] ⌈The initial events shall be transported using unicast from Server to Client.⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00488] ⌈Subscribe Eventgroup Ack entries shall reference up to 1 Multicast Option for the Internet Protocol used (IPv4 or IPv6).⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00489] ⌈The Multicast Option shall be set to UDP as transport protocol.⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00490] ⌈The client shall open the Endpoint specified in the Multicast Option referenced by the Subscribe Eventgroup Ack entry as fast as possible to not miss multicast events.⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00025)*

Example: Figure 4.15 shows an example with the different Endpoint and a Multicast Option:

- The Server offers the Service Instance on Server UDP-Endpoint SU and Server TCP-Endpoint ST
- The Client opens a TCP connection

- The Client sends a Subscribe Eventgroup entry with Client UDP-Endpoint CU (unicast) and a Client TCP-Endpoint CT.
- The Server answers with a Subscribe Eventgroup Ack entry with Multicast MU

Then the following operations happen:

- The Client calls a method on the Server
- Request is sent from CU to SU and Response from SU to CU
- For TCP this would be: Request dyn to ST and RESPONSE from ST to CT
- The Server sends a Unicast UDP Event: SU to CU
- The Server sends a Unicast TCP Event: ST to CT
- The Server sends a Multicast UDP Event: SU to MU

Keep in mind that Multicast Endpoints use a Multicast IP Address on the receiver side, i.e. the client, and TCP cannot be used for Multicast communication.

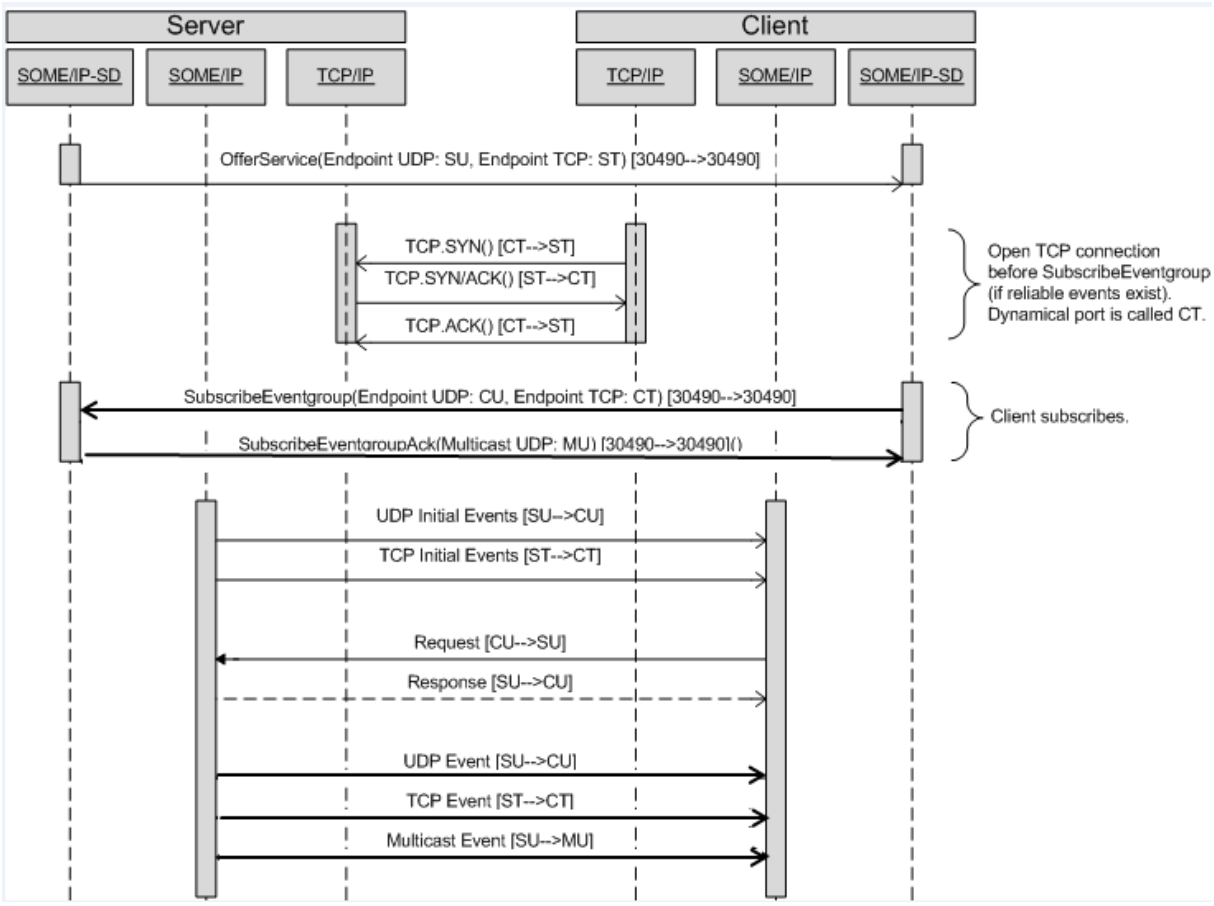

Figure 4.15: Publish/Subscribe Example for Endpoint Options and the usage of ports

4.1.3 Service Discovery Messages

[PRS_SOMEIPSD_00600] ⌈All SD Messages shall be sent to SD_PORT.⌋*(RS_SOMEIPSD_00001)*

[PRS_SOMEIPSD_00601] ⌈SD_PORT shall be used as the source port for SD Unicast/Multicast Messages.⌋*(RS_SOMEIPSD_00002, RS_SOMEIPSD_00003)*

[PRS_SOMEIPSD_00602] ⌈All unicast SD messages should have SD_PORT as destination port unless the SD Endpoint Option defines a different port.⌋*(RS_SOMEIPSD_00002)*

[PRS_SOMEIPSD_00603] ⌈All SD multicast messages shall be sent using the SD_MULTICAST_IP.⌋*(RS_SOMEIPSD_00003, RS_SOMEIPSD_00022)*

Using the previously specified header format, different entries and messages consisting of one or more entries can be built. The specific entries and their header layouts are explained in the following sections.

4.1.3.1 Eventgroup Entry

Subscribe Eventgroup Entry

[PRS_SOMEIPSD_00385] ⌈The Subscribe Eventgroup entry type shall be used to subscribe to an eventgroup.⌋*(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00386] ⌈Subscribe Eventgroup entries shall set the entry fields in the following way:

- Type shall be set to 0x06 (SubscribeEventgroup).
- Service ID shall be set to the Service ID of the service instance that includes the eventgroup subscribed to.
- Instance ID shall be set to the Instance ID of the service instance that includes the eventgroup subscribed to.
- Major Version shall be set to the Major Version of the service instance of the eventgroup subscribed to.
- Eventgroup ID shall be set to the Eventgroup ID of the eventgroup subscribed to.
- Minor Version shall be set to the Minor Version of the service instance of the eventgroup subscribed to.
- TTL shall be set to the lifetime of the subscription.
  - If set to 0xFFFFFF, the Subscribe Eventgroup entry shall be considered valid until the next reboot.

  - TTL shall not be set to 0x000000 since this is considered to be the Stop Offer Service Entry.

- Reserved shall be set to 0x00 until further notice.
- Initial Data Requested Flag shall be set to 1, if the client sends the first subscribe in sequence to trigger the sending of initial events. Set to 0 otherwise.
- Reserved2 shall be set to three 0 bits.
- Counter shall be used to differentiate between parallel subscribes to the same eventgroup of the same service (only difference in endpoint). If not used, set to 0x0.

⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00387] ⌈Subscribe Eventgroup entries shall reference one or two IPv4 and/or one or two IPv6 Endpoint Options (one for UDP, one for TCP).⌋(*RS_SOMEIPSD_00015, RS_SOMEIPSD_00025*)

Stop Subscribe Eventgroup Entry

[PRS_SOMEIPSD_00388] ⌈The Stop Subscribe Eventgroup entry type shall be used to stop subscribing to eventgroups.⌋(*RS_SOMEIPSD_00017*)

[PRS_SOMEIPSD_00389] ⌈Stop Subscribe Eventgroup entries shall set the entry fields exactly like the Subscribe Eventgroup entry they are stopping, except:

- TTL shall be set to 0x000000.

⌋(*RS_SOMEIPSD_00017*)

[PRS_SOMEIPSD_00574] ⌈A Stop Subscribe Eventgroup Entry shall reference the same options the Subscribe Eventgroup Entry referenced. This includes but is not limited to Endpoint and Configuration options.⌋(*RS_SOMEIPSD_00017*)

Subscribe Eventgroup Acknowledgement (Subscribe Eventgroup Ack) Entry

[PRS_SOMEIPSD_00390] ⌈The Subscribe Eventgroup Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was accepted.⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00391] ⌈Subscribe Eventgroup Acknowledgment entries shall set the entry fields in the following way:

- Type shall be set to 0x07 (SubscribeEventgroupAck).
- Service ID, Instance ID, Major Version, Eventgroup ID, TTL, Reserved, Initial Data Requested Flag, Reserved2, and Counter shall be the same value as in the Subscribe Eventgroup that is being answered.

⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00392] ⌈Subscribe Eventgroup Ack entries referencing events and notification events that are transported via multicast shall reference an IPv4 Multicast Option and/or and IPv6 Multicast Option.⌋(*RS_SOMEIPSD_00015*)

Subscribe Eventgroup Negative Acknowledgement (Subscribe Eventgroup Nack) Entry

[PRS_SOMEIPSD_00393] ⌈The Subscribe Eventgroup Negative Acknowledgment entry type shall be used to indicate that Subscribe Eventgroup entry was NOT accepted.⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00394] ⌈Subscribe Eventgroup Negative Acknowledgment entries shall set the entry fields in the following way:

- Type shall be set to 0x07 (SubscribeEventgroupAck).
- Service ID, Instance ID, Major Version, Eventgroup ID, Counter, and Reserved shall be the same value as in the Subscribe that is being answered.
- The TTL shall be set to 0x000000.

⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00566] ⌈Reasons to not accept a Subscribe Eventgroup include (but are not limited to):

- Combination of Service ID, Instance ID, Eventgroup ID, and Major Version is unknown
- Required TCP-connection was not opened by client
- Problems with the references options occurred
- Resource problems at the Server

⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00527] ⌈When the client receives a SubscribeEventgroupNack as answer on a SubscribeEventgroup for which a TCP connection is required, the client shall check the TCP connection and shall restart the TCP connection if needed.⌋(*RS_SOMEIPSD_00015*)

Rational:
The server might have lost the TCP connection and the client has not.

Checking the TCP connection might include the following:

- Checking whether data is received for e.g. other Eventgroups.
- Sending out a Magic Cookie message and waiting for the TCP ACK.

- Reestablishing the TCP connection.

4.1.4 Service Discovery Communication Behavior

[PRS_SOMEIPSD_00800] ⌈SOME/IP Service Discovery shall reduce the number of Service Discovery messages by packing entries together, if they can be sent at the same time. E.g.:

- Multiple entries of different service instances
- Mulitple entries of different types (e.g. Offer entries, answers of Subscribed EventGroup entries ... a.s.o.)

⌋*(RS_SOMEIPSD_00025)*

4.1.4.1 Startup Behavior

[PRS_SOMEIPSD_00395] ⌈For each Service Instance the Service Discovery shall have at least these three phases in regard to sending entries:

- Initial Wait Phase
- Repetition Phase
- Main Phase

⌋*(RS_SOMEIPSD_00025)*

Note:
An actual implemented state machine will need more than just states for these three phases. E.g. local services can be still down and remote services can be already known (no finds needed anymore).

[PRS_SOMEIPSD_00397] ⌈The service discovery shall enter the Initial Wait Phase for a client service instance when the link on the interface needed for this service instance is up and the client service is requested by the application.⌋*(RS_SOMEIPSD_00025, RS_SOMEIPSD_00012)*

[PRS_SOMEIPSD_00133] ⌈The service discovery shall enter the Initial Wait Phase for a server service instance when the link on the interface needed for this service instance is up and the server service is available.⌋*(RS_SOMEIPSD_00025, RS_SOMEIPSD_00012)*

Note:
It is possible that the link is up but the service is not yet available on server side

Systems has started means here the needed applications and possible external sensors and actuators as well. Basically the functionality needed by this service instance

has to be ready to offer a service and finding a service is applicable after some application requires it.

[PRS_SOMEIPSD_00399] ⌈The Service Discovery shall wait based on the INITIAL_DELAY after entering the Initial Wait Phase and before sending the first messages for the Service Instance.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00400] ⌈INITIAL_DELAY shall be defined as a minimum and a maximum delay.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00401] ⌈The wait time shall be determined by choosing a random value between the minimum and maximum of INITIAL_DELAY.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00804] ⌈The Service Discovery shall use the same random value, if ClientService and ServerService reference the same ClientServiceTimer and ServerServiceTimer, respectively, and if it its ensured that the referencing ClientServiceS and ServerServiceS, respectively, are requested and released in the same point in time.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00805] ⌈The Service Discovery shall use different random values per ClientService and ServerService, if the ClientServices and ServerService referencing their own ClientServiceTimer and ServerServiceTimer, respectively. Thus, if a ClientService or ServerSerivce enters the Initial Wait Phase, they shall use an individual calculated random value within the Initial Wait Phase.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00404] ⌈After sending the first message the Repetition Phase of this Service Instance/these Service Instances is entered.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00405] ⌈The Service Discovery shall wait in the Repetitions Phase based on REPETITIONS_BASE_DELAY.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00406] ⌈After each message sent in the Repetition Phase the delay is doubled.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00407] ⌈The Service Discovery shall send out only up to REPETITIONS_MAX entries during the Repetition Phase.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00408] ⌈Sending Find entries shall be stopped after receiving the corresponding Offer entries by jumping to the Main Phase in which no Find entries are sent.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00409] ⌈If REPETITIONS_MAX is set to 0, the Repetition Phase shall be skipped and the Main Phase is entered for the Service Instance after the Initial Wait Phase.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00410] ⌈After the Repetition Phase the Main Phase is being entered for a Service Instance.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00411] ⌈After entering the Main Phase, the provider shall wait 1*CYCLIC_OFFER_DELAY before sending the first offer entry message.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00412] ⌈In the Main Phase Offer Messages shall be sent cyclically if a CYCLIC_OFFER_DELAY is configured.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00413] ⌈After a message for a specific Service Instance the Service Discovery waits for 1*CYCLIC_OFFER_DELAY before sending the next message for this Service Instance.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00415] ⌈For Find entries (Find Service and Find Eventgroup) no cyclic messages are allowed in Main Phase.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00582] ⌈Subscribe EventGroup Entries shall be triggered by Offer entries, which are sent cyclically.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00416] ⌈Example:

Initial Wait Phase:

- Wait for random_delay in Range(INITIAL_DELAY_MIN, _MAX)
- Send message (Find Service and Offer Service entries)

Repetition Phase (REPETITIONS_BASE_DELAY=100ms, REPETITIONS_MAX=2):

- Wait $2^0 * 100ms$
- Send message (Find Service and Offer Service entries)
- Wait $2^1 * 100ms$
- Send message (Find Service and Offer Service entries)

Main Phase (as long message is active and CYCLIC_OFFER_DELAY is defined):

- Wait CYCLIC_OFFER_DELAY
- Send message (Offer Service entries)

⌋*(RS_SOMEIPSD_00025)*

4.1.4.2 Server Answer Behavior

[PRS_SOMEIPSD_00417] ⌈The Service Discovery shall delay answers to entries that were received in multicast SOME/IP-SD messages using the configuration item REQUEST_RESPONSE_DELAY. This is valid for the following two cases:

- Offer entry (unicast or multicast) after received find entry (multicast)
- Subscribe entry (unicast) after received offer entry (multicast)

⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00419] ⌈The REQUEST_RESPONSE_DELAY shall not apply if unicast messages are answered with unicast messages.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00420] ⌈REQUEST_RESPONSE_DELAY shall be specified by a minimum and a maximum.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00421] ⌈The actual delay shall be randomly chosen between minimum and maximum of REQUEST_RESPONSE_DELAY.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00422] ⌈For basic implementations all Find Service entries (no matter of the state of the Unicast Flag) shall be answered with Offer Service entries transported using unicast.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00423] ⌈For optimization purpose the following behaviors shall be supported as option:

- Find messages received with the Unicast Flag set to 1 in main phase, shall be answered with a unicast response if the last offer was sent less than `1/2 CYCLIC_OFFER_DELAY` ago.
- Find messages received with the Unicast Flag set to 1 in main phase, shall be answered with a multicast RESPONSE if the last offer was sent `1/2 CYCLIC_OFFER_DELAY` or longer ago.
- Find messages received with Unicast Flag set to 0 (multicast), shall be answered with a multicast response. (Note: this was only needed in earlier migration scenarios and will go away in the future)

⌋*(RS_SOMEIPSD_00025)*

4.1.4.3 Shutdown Behavior

[PRS_SOMEIPSD_00427] ⌈When a server service instance of an ECU is in the Repetition and Main Phase and is being stopped, a Stop Offer Service entry shall be sent out.⌋*(RS_SOMEIPSD_00017, RS_SOMEIPSD_00012)*

[PRS_SOMEIPSD_00751] ⌈When the link goes down for a server service instance in the Initial Wait Phase, Repetition Phase or Main Phase, the service discovery shall enter the Initial Wait Phase when the link is up again and the service is still available.⌋*(RS_SOMEIPSD_00017)*

[PRS_SOMEIPSD_00752] ⌈When the link goes down for a client service instance in the Initial Wait Phase, Repetition Phase or Main Phase, the service discovery shall enter the Initial Wait Phase when the link is up again and the service is still requested.⌋*(RS_SOMEIPSD_00017)*

[PRS_SOMEIPSD_00428] ⌈When a server sends out a Stop Offer Service entry all subscriptions for this service instance shall be deleted on the server side.⌋*(RS_SOMEIPSD_00017)*

[PRS_SOMEIPSD_00429] ⌈When a client receives a Stop Offer Service entry all subscriptions for this service instance shall be deleted on the client side.⌋*(RS_SOMEIPSD_00017)*

[PRS_SOMEIPSD_00430] ⌈When a client receives a Stop Offer Service entry, the client shall not send out Find Service entries but wait for Offer Service entry or change of status (application, network management, Ethernet link, or similar).⌋*(RS_SOMEIPSD_00017)*

[PRS_SOMEIPSD_00431] ⌈When a client service instance of an ECU is in the Main Phase and is being stopped (i.e. the service instance is released), the SD shall send out Stop Subscribe Eventgroup entries for all subscribed Eventgroups.⌋*(RS_SOMEIPSD_00017)*

[PRS_SOMEIPSD_00432] ⌈When the whole ECUs is being shut down Stop Offer Service entries shall be sent out for all service entries and Stop Subscribe Eventgroup entries for Eventgroups.⌋*(RS_SOMEIPSD_00017)*

4.1.4.4 State Machines

[PRS_SOMEIPSD_00433] ⌈In this section the state machines of the client and server are shown.⌋*(RS_SOMEIPSD_00025)*

[PRS_SOMEIPSD_00434] ⌈SOME/IP Service State Machine Server is described as follows:

States inside `SD Server State Machine(Service)` are defined as follows:

- `SD Server State Machine(Service)`
  - `Not Ready`
  - `Ready`
    - `Initial Wait Phase`
      - `Timer Set`
    - `Repetition Phase`
      - `Timer Set`
    - `Main Phase`
      - `Timer Set`

Initial entry points of `SD Server State Machine(Service)` are inside the following states:

- `SD Server State Machine(Service)`
  - `Ready`
    - `Initial Wait Phase`
    - `Repetition Phase`
    - `Main Phase`

Transitions inside `SD Server State Machine(Service)` are defined as follows:

FROM entry point `SD Server State Machine(Service)`
TO `Not Ready`
WITH `[ifstatus!=up_and_configured or service-status==down]`

FROM entry point `SD Server State Machine(Service)`
TO `Not Ready`
WITH `[ifstatus==up_and_configured or service-status==up]`

FROM `Not Ready`
TO `Ready`
WITH `if-status-changed() or service-status-changed() [ifstatus==up_and_configured and service-status==up]`

FROM `Ready`
TO `Not Ready`
WITH `if-status-changed [ifstatus!=up_and_configured] /clearAllTimers()`

FROM `Ready`
TO `Not Ready`
WITH `service-status==down /clearAllTimers() send(StopOfferService)`

FROM `TimerSet`
OF `Initial Wait Phase`
TO `Repetition Phase`
WITH `Timer expired /send(OfferService)`

FROM `TimerSet`
OF `Repetition Phase`
TO `TimerSet`
OF `Repetition Phase`
WITH `receive(FindService) /waitAndSend(OfferService) ResetTimer()`

FROM `TimerSet`
OF `Repetition Phase`
TO `TimerSet`
OF `Repetition Phase`
WITH `Timer expired [run<REPETITIONS_MAX] /send(OfferService) run++ setTimer((2^run)*REPETITIONS_BASE_DELAY`

FROM `TimerSet`
OF `Repetition Phase`
TO `Main Phase`
WITH `Timer expired [run==REPETITIONS_MAX]`

FROM entry point `Ready`
TO `Initial Wait Phase`

FROM entry point `Initial Wait Phase`
TO `Timer Set`
OF `Initial Wait Phase`
WITH `SetTimerInRange(INITIAL_DELAY_MIN,INITIAL_DELAY_MAX)`

FROM entry point `Repetition Phase`
TO `Timer Set`
OF `Repetition Phase`
WITH `[REPETITIONS_MAX>0] /run=0 setTimer((2^run)*REPETITIONS_BASE_DELAY)`

FROM entry point `Repetition Phase`
TO `Main Phase`
WITH `[REPETITIONS_MAX==0]`

FROM entry point `Main Phase`
TO `Timer Set`
OF `Main Phase`
WITH `/setTimer(CYCLIC_ANNOUNCE_DELAY) send(OfferService)`

FROM `Timer Set`
OF `Main Phase`
TO `Timer Set`
OF `Main Phase`
WITH `Timer expired /setTimer(CYCLIC_ANNOUNCE_DELAY) send(OfferService)`

FROM `Timer Set`
OF `Main Phase`
TO `Timer Set`
OF `Main Phase`
WITH `receive(FindService) /waitAndSend(OfferService) resetTimer()`

⌋(*RS_SOMEIPSD_00025*)

Note: Graphical information of the SOME/IP Service State Machine Server is shown in Figure 4.16

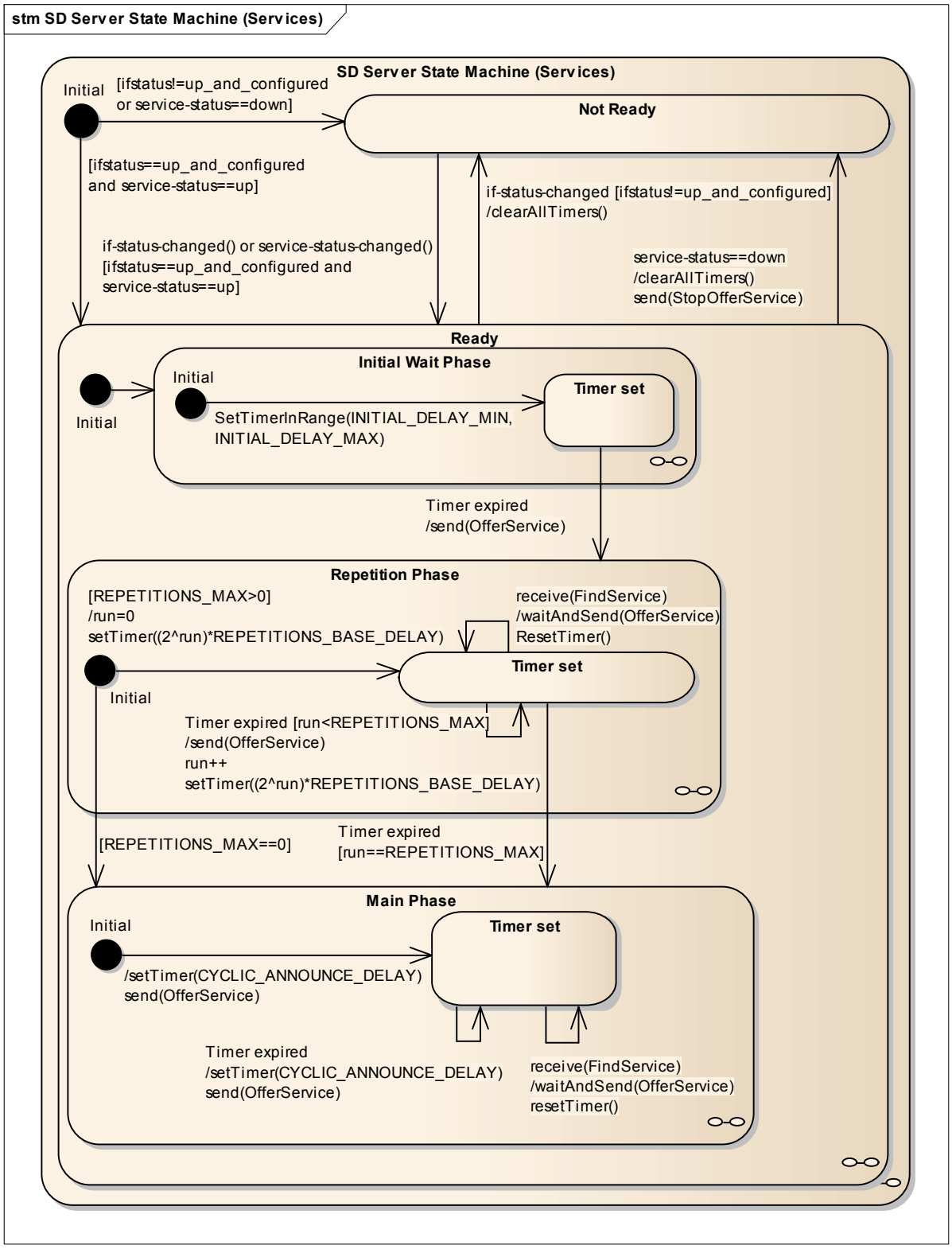

Figure 4.16: SOME/IP Service State Machine Server

[PRS_SOMEIPSD_00435] ⌈SOME/IP Service State Machine Client is described as follows:

States inside `SD Client State Machine(Service)` are defined as follows:

- `SD Client State Machine(Service)`
  - `Not Requested`
    - `Service Not Seen`
    - `Service Seen`
  - `Requested_but_not_ready`
  - `Main`
    - `Service Ready`
    - `Stopped`
  - `Searching for Service`
    - `Initial Wait Phase`
      - `Timer Set`
    - `Repetition Phase`
      - `Timer Set`

Initial entry points of `SD Client State Machine(Service)` are inside the following states:

- `SD Client State Machine(Service)`
  - `Not Requested`
- `Searching for Service`
  - `Initial Wait Phase`
  - `Repetition Phase`

Transitions inside `SD Client State Machine(Service)` are defined as follows:

FROM entry point `SD Client State Machine(Service)`
TO `Not Requested`
WITH `[Service Not Requested]`

FROM entry point `SD Client State Machine(Service)`
TO `Requested_but_not_ready`
WITH `Service Not Requested and ifstatus!=up_and_configured`

FROM entry point `SD Client State Machine(Service)`
TO `Searching for Service`
WITH `Service Requested and ifstatus==up_and_configured`

FROM entry point `Not Requested` TO `Service Not Seen`

FROM `Not Requested` TO `Requested_but_not_ready`
WITH `InternalServiceRequest [ifstatus!=up_and_configured]`

FROM `Service Not Seen`
TO `Service Seen`
WITH `receive(OfferService) /setTimer(TTL)`

FROM `Repetition Phase`
TO `Stopped`
WITH `Repetition Expired`

FROM `Stopped`
TO `Service Not Seen`
WITH `[ServiceNotRequired]`

FROM `Service Seen`
TO `Service Not Seen`
WITH `if-status-changed() [ifstatus!=up_and_configured]`

FROM `Service Seen`
TO `Service Not Seen`
WITH `Timer expired (TTL)`

FROM `Service Seen`
TO `Service Not Seen`
WITH `receive(StopServiceOffer)`

FROM `Service Seen`
TO `Service Seen`

FROM `Service Seen`
TO `Service Ready`
WITH `InternalServiceRequest [ifstatus==up_and_configured]`

FROM `Service Ready`
TO `Service Seen`
WITH `[ServiceNotRequest]`

FROM `Service Ready`
TO `Service Ready`
WITH `receive(OfferService) /resetTimer(TTL)`

FROM `Service Ready`
TO `Stopped`
WITH `receive(OfferService) /chancelTimer(TTL)`

FROM `Stopped`
TO `Service Ready`
WITH `receive(OfferService) /resetTimer(TTL)`

FROM `Service Ready`
TO `Searching for Service`
WITH `Timer expired (TTL)`

FROM `Searching for Service`
TO `Service Ready`
WITH `receive(OfferService) /setTimer(TTL)`

FROM `Searching for Service`
TO `Requested_but_not_ready`
WITH `if-status-changed() [ifstatus!=up_and_configured] /cancel-Timer(TTL)`

FROM `Requested_but_not_ready`
TO `Searching for Service`
WITH `if-status-changed() [ifstatus!=up_and_configured]`

FROM entry point `Searching for Service`
TO `Initial Wait Phase`

FROM entry point `Initial Wait Phase`
TO `Timer Set`
OF `Initial Wait Phase`

WITH `/setTimerInRange(INITIAL_DELAY_MIN, INITIAL_DELAY_MAX)`

FROM `Timer Set`
OF `Initial Wait Phase`
TO `Repetition Phase`
WITH `TimerExpired /send(FindService)`

FROM entry point `Repetition Phase`
TO `Timer Set`
OF `Repetition Phase`
WITH `[REPETITONS_MAX>0] /run=0 setTimer((2^run)*REPETITIONS_BASE_DELAY)`

FROM `Timer Set`
OF `Repetition Phase`
TO `Timer Set`
OF `Repetition Phase`

FROM `Not Requested`
TO `Requested_but_not_ready`
WITH `InternalServiceRequest [ifstatus!=up_and_configured]`

⌋(*RS_SOMEIPSD_00025*)

Note: Graphical information of the SOME/IP Service State Machine Client is shown in Figure 4.17

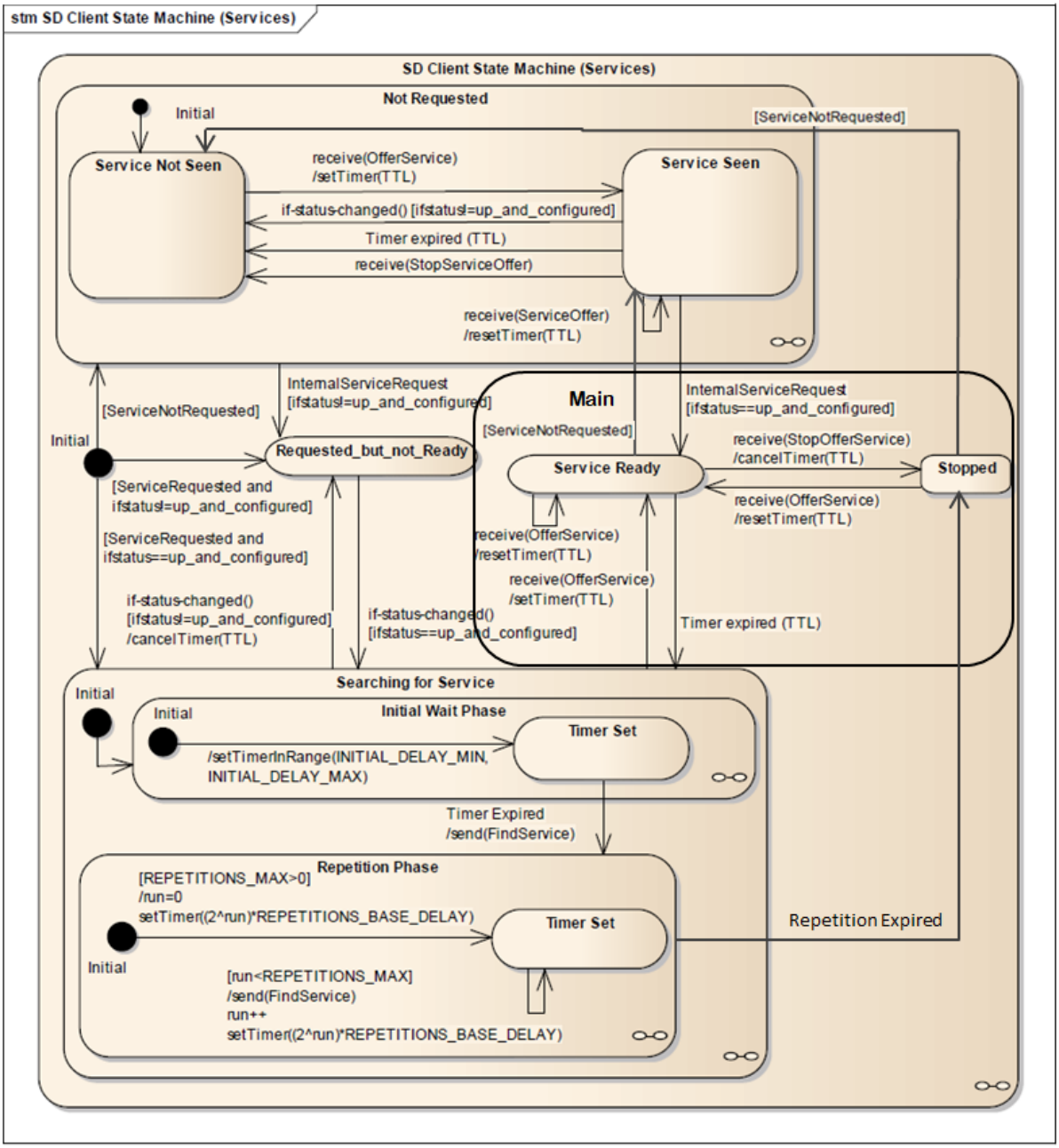

Figure 4.17: SOME/IP Service State Machine Client

4.1.4.5 SOME/IP-SD Mechanisms and Errors

In this section SOME/IP-SD in cases of errors (e.g. lost or corrupted packets) is discussed. This is also be understood as rationale for the mechanisms used and the configuration possible.

Soft State Protocol: SOME/IP-SD was designed as soft state protocol, that means that entries come with a lifetime and need to be refreshed to stay valid (setting the TTL to the maximum value shall turn this off).

Initial Wait Phase:

The Initial Wait Phase was introduced for two reasons: deskewing events of starting ECUs to avoid traffic bursts and allowing ECUs to collect multiple entries in SD messages.

Repetition Phase:

The Repetition Phase was introduced to allow for fast synchronization of clients and servers. If the clients startup later, it will find the server very fast. And if the server starts up later, the client can be found very fast. The Repetition Phase increases the time between two messages exponentially to avoid that overload situtaions keep the system from synchronization.

Main Phase:

In the Main Phase the SD tries to stabilize the state and thus decreases the rate of packets by sending no Find Services anymore and only offers in the cyclic interval (e.g. every 1s).

Request-Response-Delay:

SOME/IP-SD shall be configured to delay the answer to entries in multicast messages by the Request-Response-Delay. This is useful in large systems with many ECUs. When sending a SD message with many entries in it, a lot of answers from different ECUs arrive at the same time and put large stress on the ECU receiving all these answers.

4.1.4.6 Error Handling

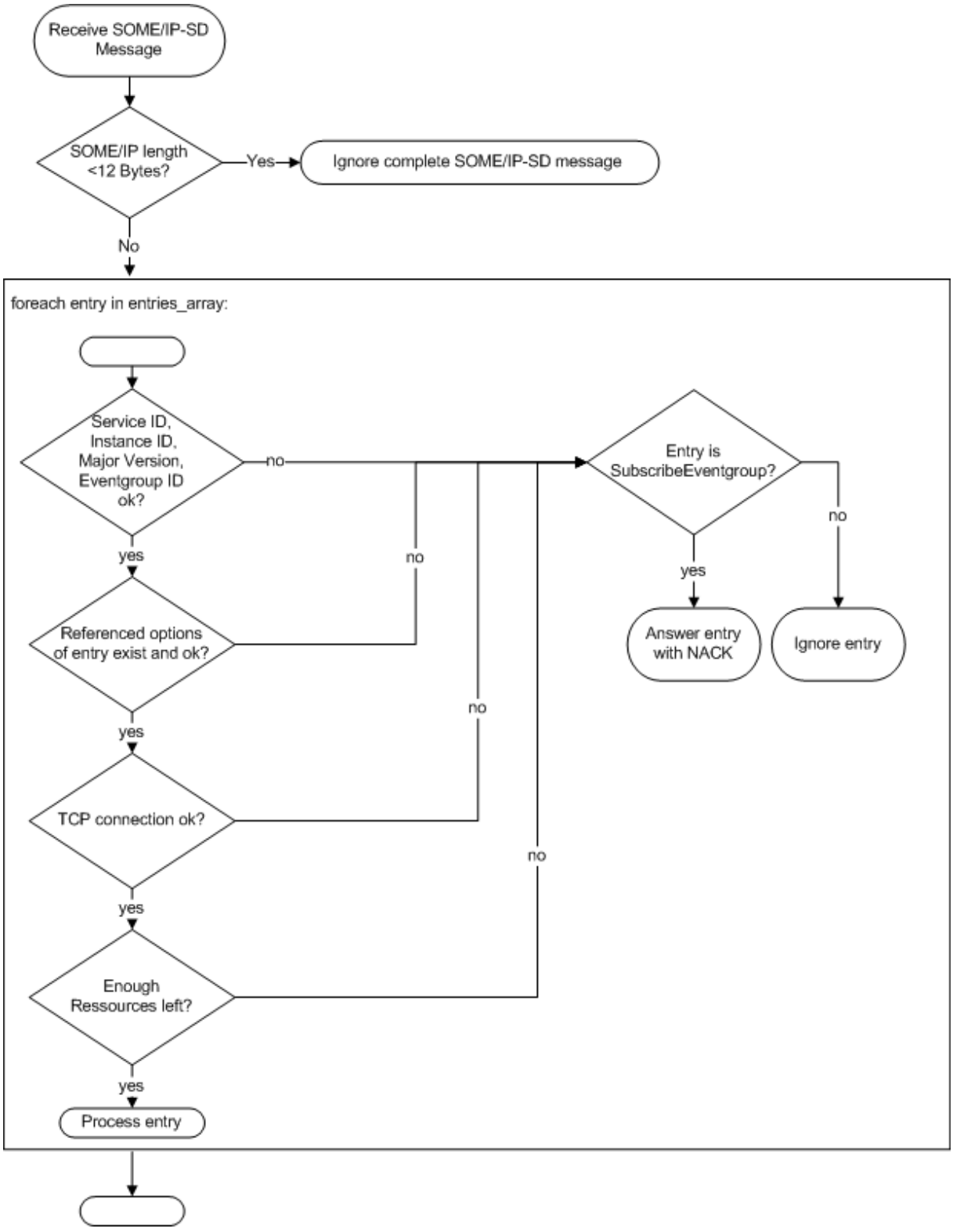

Figure 4.18: SOME/IP-SD Error Handling

Figure 4.18 shows a simplified process for the error handling of incoming SOME/IP-SD messages.

[PRS_SOMEIPSD_00125] ⌈Check that at least enough bytes for an empty SOME/IP-SD message are present, i.e the message is at least 12 Bytes long. If the check fails, the message shall be discarded without further actions.⌋*(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00126] ⌈If the Service ID of a received entry is not known, the entry shall be ignored.⌋*(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00127] ⌈If the Instance ID of a received entry is not known, the entry shall be ignored.⌋*(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00128] ⌈If the Major Version of a received entry is not known, the entry shall be ignored.⌋*(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00129] ⌈If the Eventgroup ID of a received entry is not known, the entry shall be ignored. This is only applicable to eventgroup entries.⌋*(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00803] ⌈If the length of the Entries Array has an invalid size (i.e. the entries array would exceed the message size), the message shall be discarded without further actions.⌋*(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00130] ⌈Check the referenced Options of each received entry:

- The referenced options exist.
- The entry references all required options (e.g. a provided eventgroup that uses unicast requires a unicast endpoint option in a received Subscribe Eventgroup entry).
- The entry only references supported options (e.g. a required eventgroup that does not support multicast data reception does not support multicast endpoint options in a Subscribe Eventgroup ACK entry).
- There are no conflicts between the options referenced by an entry (i.e. two options of same type with contradicting content).
- The Type of the referenced Option is known or the discardable flag is set to 1.
- The Type of the referenced Option is allowed for the entry [PRS_SOMEIPSD_00583] or discardable flag is set to 1.
- The Length of the referenced Option is consistent to the Type of the Option.
- An Endpoint Option has a valid L4-Protocol field.
- The Option is valid (e.g. a multicast endpoint option shall use a multicast IP address).

⌋*(RS_SOMEIPSD_00019)*

Note:
If an entry references an option that is known by the Service Discovery implementation but not required by the service (e.g. an Offer references a TCP and UDP option and the client uses only UDP, or a Subscribe Eventgroup entry references a UDP endpoint option but the server uses only multicast event transmission), the entry shall be processed.

[PRS_SOMEIPSD_00131] ⌈Check if the TCP connection is already present (only applicable, if TCP is configured for Eventgroup and Subscribe Eventgroup entry was received)⌋ *(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00132] ⌈Check if enough resources are left (e.g. Socket Connections)⌋ *(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00232] ⌈If the checks in [PRS_SOMEIPSD_00130] fail for a received Find entry, the entry shall be ignored.⌋ *(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00233] ⌈If the checks in [PRS_SOMEIPSD_00130] fail for a received Offer entry, the entry shall be ignored.⌋ *(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00234] ⌈If the checks in [PRS_SOMEIPSD_00130], [PRS_SOMEIPSD_00131], or [PRS_SOMEIPSD_00132] fail for a received Subscribe Eventgroup entry, a Subscribe Eventgroup NACK entry shall be sent.⌋ *(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00235] ⌈If the checks in [PRS_SOMEIPSD_00130] or [PRS_SOMEIPSD_00132] fail for a received Subscribe Eventgroup ACK entry, the entry shall be processed, but the subscription shall not be considered as successful.⌋ *(RS_SOMEIPSD_00019)*

[PRS_SOMEIPSD_00231] ⌈Options that are referenced by an entry shall be ignored if:

- The Option Type is not known (i.e. not yet specified, or not supported by the receiver) and the discardable flag is set to 1.
- The option is redundant (i.e. another option of the same type and same content is referenced by this entry).
- The option is not required (e.g. a provided eventgroup that uses only multicast does not require a unicast endpoint option in a received Subscribe Eventgroup entry, though it is still allowed).

⌋ *(RS_SOMEIPSD_00019)*

4.1.5 Announcing non-SOME/IP protocols with SOME/IP-SD

Besides SOME/IP other communication protocols are used within the vehicle; e.g. for Network Management, Diagnosis, or Flash Updates. Such communication protocols might need to communicate a service instance or have eventgroups as well.

[PRS_SOMEIPSD_00437] ⌈For Non-SOME/IP protocols (the application protocol itself doesn't use SOME/IP but it is published over SOME/IP SD) a special Service-ID shall be used and further information shall be added using the configuration option:

- Service-ID shall be set to 0xFFFE (reserved)
- Instance-ID shall be used as described for SOME/IP services and eventgroups.
- The Configuration Option shall be added and shall contain exactly one entry with key "otherserv" and a configurable non-empty value that is determined by the system department.

⌋*(RS_SOMEIPSD_00004)*

[PRS_SOMEIPSD_00438] ⌈SOME/IP services shall not use the otherserv-string in the Configuration Option.⌋*(RS_SOMEIPSD_00004)*

[PRS_SOMEIPSD_00439] ⌈For Find Service/Offer Service/Request Service entries the otherserv-String shall be used when announcing non-SOME/IP service instances.⌋ *(RS_SOMEIPSD_00004)*

[PRS_SOMEIPSD_00440] ⌈

Example for valid otherserv-string: "otherserv=internaldiag".
Example for an invalid otherserv-string: "otherserv".
Example for an invalid otherserv-string: "otherserv=".⌋*(RS_SOMEIPSD_00004)*

Example:

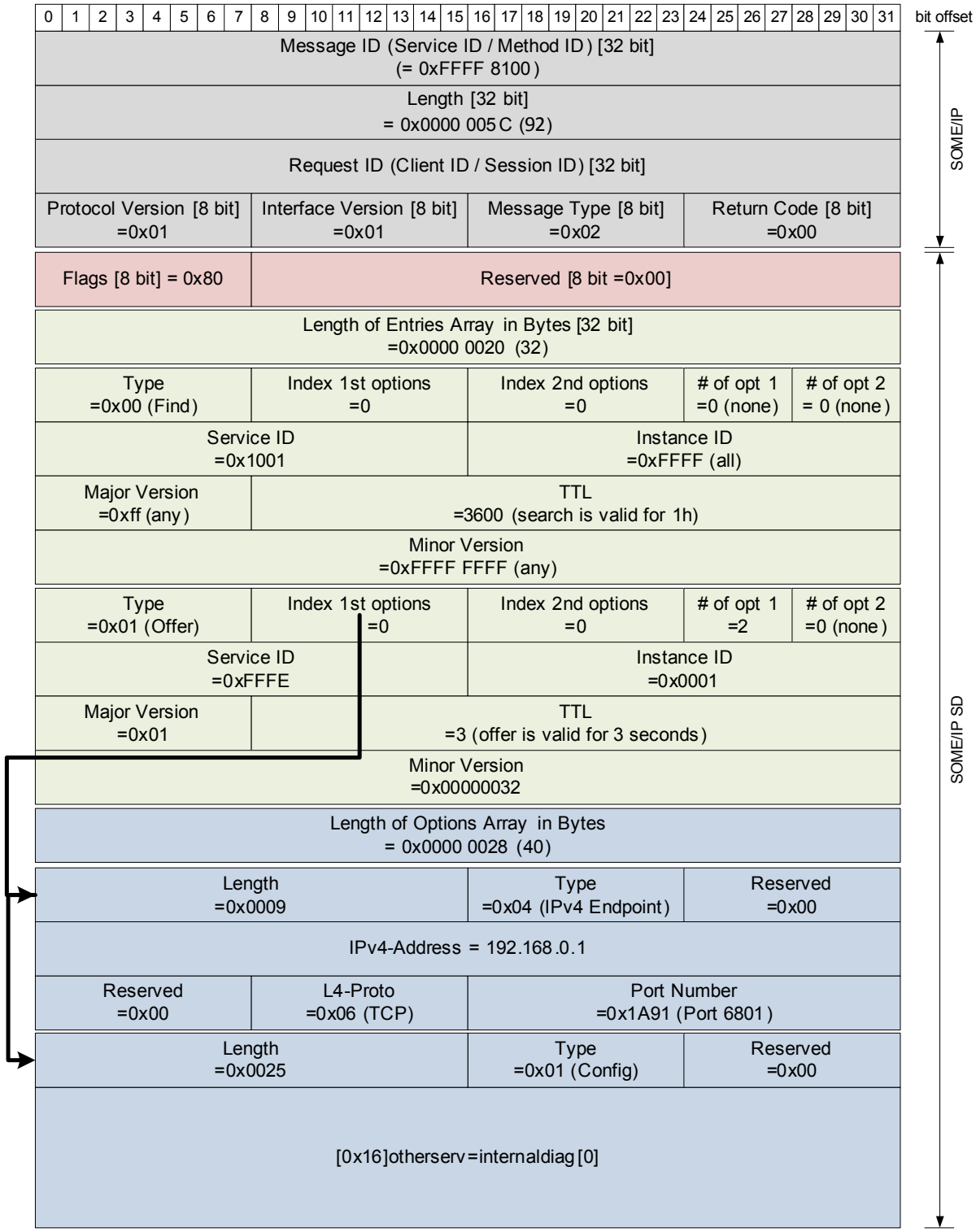

Figure 4.19: SOME/IP-SD Example PDU for Non-SOME/IP-SD

4.1.6 Publish/Subscribe with SOME/IP and SOME/IP-SD

Note: In contrast to the SOME/IP request/response mechanism there may be cases in which a client requires a set of parameters from a server, but does not want to request that information each time it is required. These are called notifications and concern events and fields.

[PRS_SOMEIPSD_00443] ⌈All clients needing events and/or notification events shall register using the SOME/IP-SD at run-time with a server.⌋ *(RS_SOMEIPSD_00013, RS_SOMEIPSD_00014, RS_SOMEIPSD_00015, RS_SOMEIPSD_00016)*

The Notification Interaction sequence is as shown below.

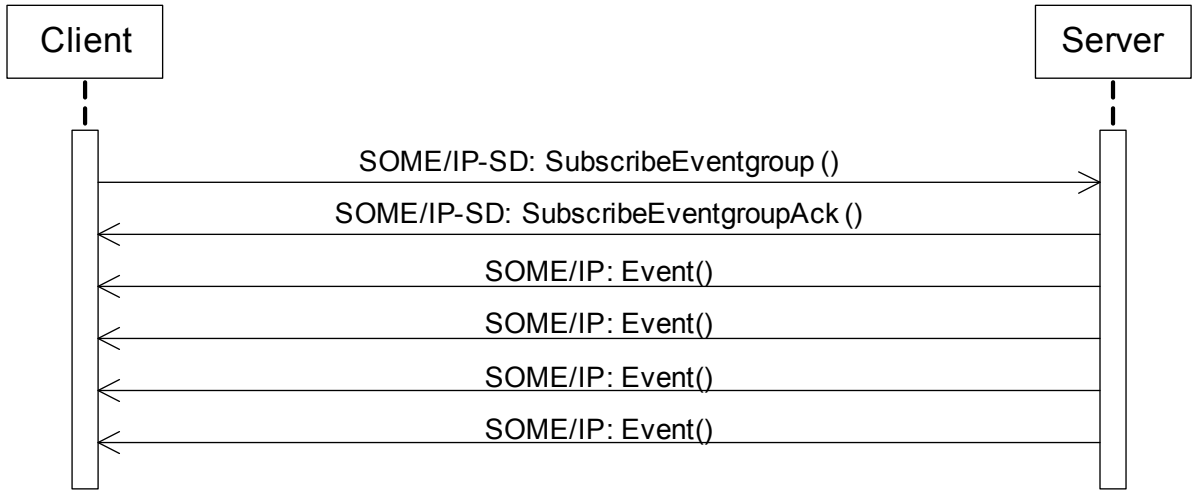

Figure 4.20: Notification interaction

This feature is comparable but NOT identical to the MOST notification mechanism.

[PRS_SOMEIPSD_00446] ⌈With the SOME/IP-SD entry Offer Service the server offers to push notifications to clients; thus, it shall be used as trigger for Subscriptions.⌋ *(RS_SOMEIPSD_00013, RS_SOMEIPSD_00014, RS_SOMEIPSD_00015, RS_SOMEIPSD_00016)*

[PRS_SOMEIPSD_00449] ⌈Each client shall respond to a SOME/IP-SD Offer Service entry from the server with a SOME/IP-SD Subscribe Eventgroup entry as long as the client is still interested in receiving the notifications/events of this eventgroup.

If the client is able to reliably detect the reboot of the server using the SOME/IP-SD messages reboot flag, the client may choose to only answer Offer Service messages after the server reboots if configured to do so (TTL set to maximum value). The client make sure that this works reliable even when the SOME/IP-SD messages of the server are lost.⌋ *(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00703] ⌈The client shall explicitly request Initial Events by setting the Initial Data Requested Flag, if it has no active subscription to the Eventgroup.⌋ *(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00704] ⌈If the client sends out additional Subscribe Eventgroup entries and the TTL of the previous Subscribe has not expired, then the client shall not request Initial Events.⌋ *(RS_SOMEIPSD_00015, RS_SOMEIPSD_00020)*

Reasons for the client to explicitly request Initial Events include but are not limited to:

- The client is currently not subscribed to the Eventgroup.
- The client has seen a link-down/link-up after the last Subscribe Eventgroup entry.
- The client has not received a Subscribe Eventgroup Ack after the last regular Subscribe Eventgroup
- The client has detected a Reboot of the Server of this Services

[PRS_SOMEIPSD_00570] ⌈If the client subscribes to two or more eventgroups including one or more identical events or fields, the server shall not send duplicated events or notification events for the field. This does mean regular events and not initial events.⌋ *(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00450] ⌈Publish/Subscribe with link loss at client side is described as follows:

1. **No prior registrations + Client subscribes**
   (a) Server: OfferService()
   (b) Client: SubscribeEventgroup**[Session ID=x, Reboot=0]**
   (c) Server: updateRegistration()
   (d) Server: SubscribeEventgroupAck + Events()
2. **Link loss at client side**
   (a) Client: linkDown()
   (b) Client: deleteEntries()
   (c) Client: linkUp()
3. **Client subscribes again, Client Reboot detected**
   (a) Server: OfferService()
   (b) Client: SubscribeEventgroup**[Session ID=1, Reboot=1]**
   (c) Server: updateRegistration()
   (d) Server SubscribeEventgroupAck + Events()

⌋ *(RS_SOMEIPSD_00015)*

Note: Description is also shown in Figure 4.21.

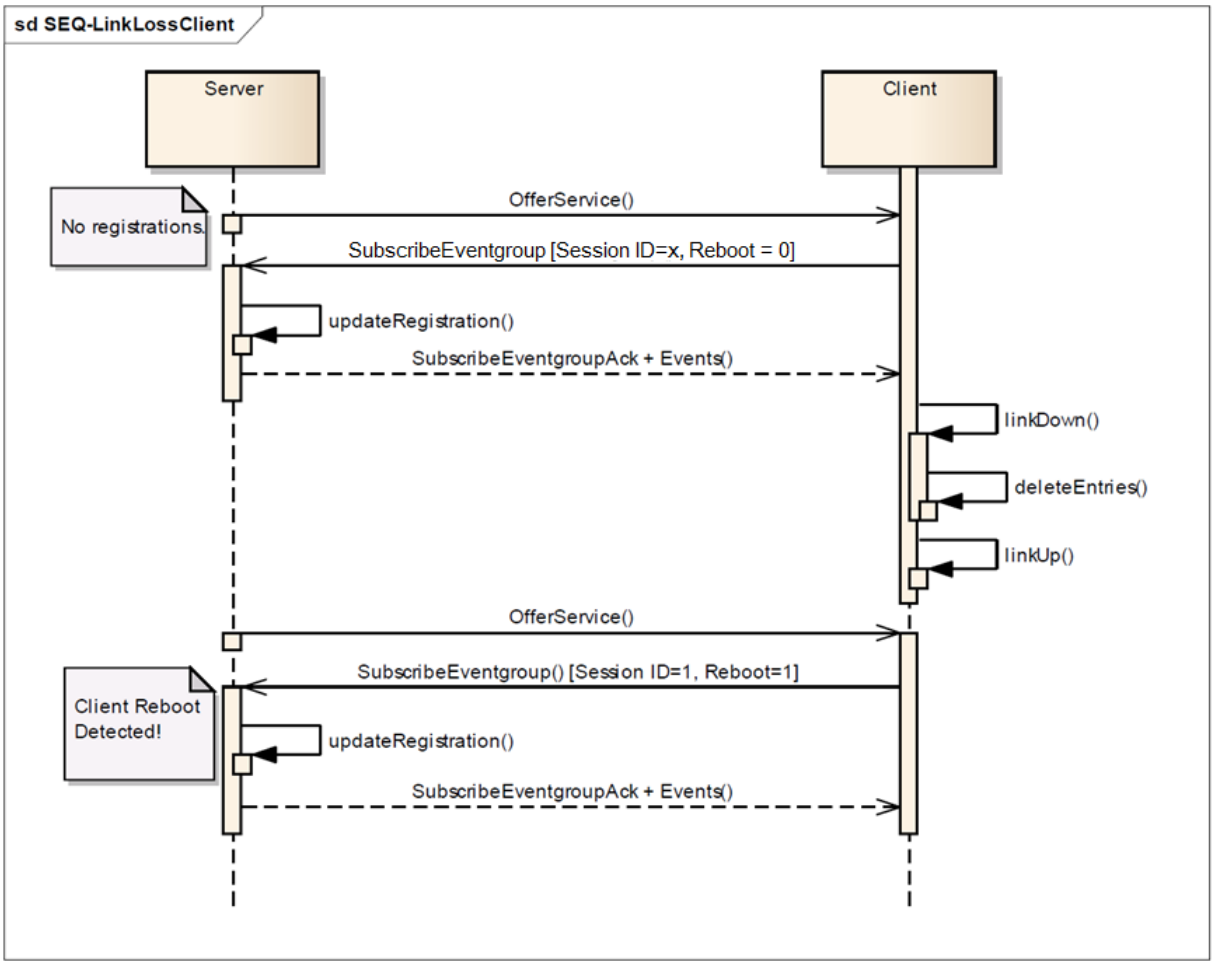

Figure 4.21: Publish/Subscribe with link loss at client (figure ignoring timings)

Note: The server sending Offer Service entries as implicit Publishes has to keep state of Subscribe Eventgroup messages for this eventgroup instance in order to know if notifications/events have to be sent.

[PRS_SOMEIPSD_00452] ⌈A client shall deregister from a server by sending a SOME/IP-SD Subscribe Eventgroup message with TTL=0 (Stop Subscribe Eventgroup see [PRS_SOMEIPSD_00389]).⌋ *(RS_SOMEIPSD_00017, RS_SOMEIPSD_00020)*

[PRS_SOMEIPSD_00453] ⌈

Publish/Subscribe Registration/Deregistration behavior is described as follows:

1. **Client 1 subscribes**
   (a) Server: OfferService() to Client 1 and Client 2
   (b) Client 1: SubscribeEventgroup()
   (c) Server: updateRegistration()
   (d) Server: SubscribeEventgroupAck + Events() to Client 1
2. **Client 2 subscribes**

(a) Client 2: SubscribeEventgroup()

(b) Server: updateRegistration()

(c) Server: SubscribeEventgroupAck + Events() to Client 2

3. **Client 2 stops subscription**

(a) Client 2: StopSubscribeEventgroup()

(b) Server: updateRegistration()

4. **Client 1 remains registered**

⌋(*RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00017*)

Note: Description is also shown in Figure 4.22.

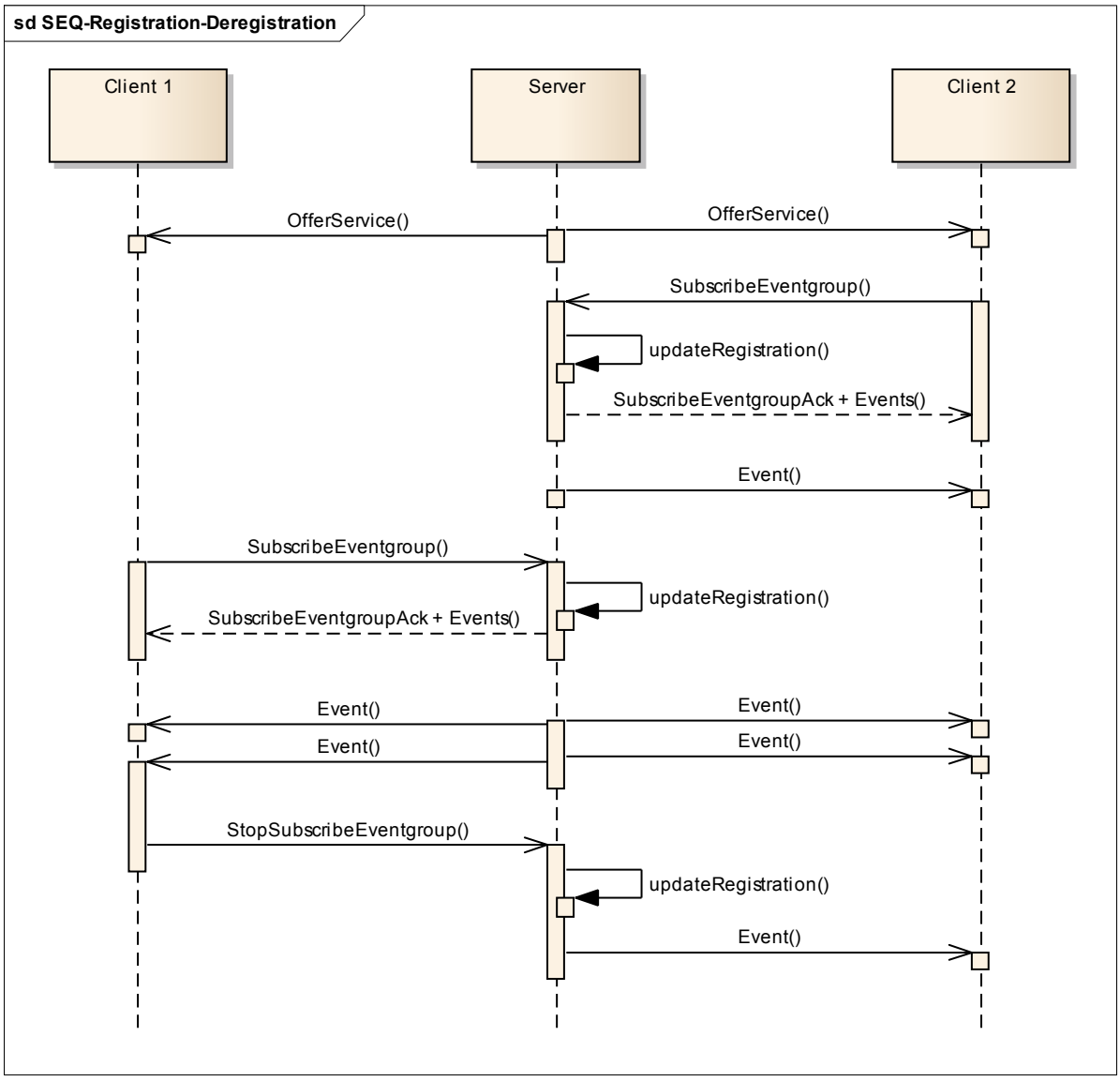

Figure 4.22: Publish/Subscribe Registration/Deregistration behavior (figure ignoring timings)

[PRS_SOMEIPSD_00454] ⌈The SOME/IP-SD on the server shall delete the subscription, if a relevant SOME/IP error occurs after sending an event or notification event.⌋ *(RS_SOMEIPSD_00017, RS_SOMEIPSD_00019)*

The error includes but is not limited to not being able to reach the communication partner and errors of the TCP connection.

[PRS_SOMEIPSD_00457] ⌈

Publish/Subscribe with link loss at server isis described as follows:

1. **No prior registrations + Client subscribes**
   (a) Server: OfferService()
   (b) Client: SubscribeEventgroup()
   (c) Server: updateRegistration()
   (d) Server: SubscribeEventgroupAck + Events()
2. **Link loss at server side**
   (a) Client: linkDown()
   (b) Client: deleteRegistrations()
   (c) Client: linkUp()
3. **Server offers again, Server Reboot detected by client**
   (a) Server: OfferService()**[Session ID=1, Reboot=1]**
   (b) Client: SubscribeEventgroup()
   (c) Server: updateRegistration()
   (d) Server SubscribeEventgroupAck + Events()

⌋*(RS_SOMEIPSD_00013, RS_SOMEIPSD_00015)*

Note: Description is also shown in Figure 4.23

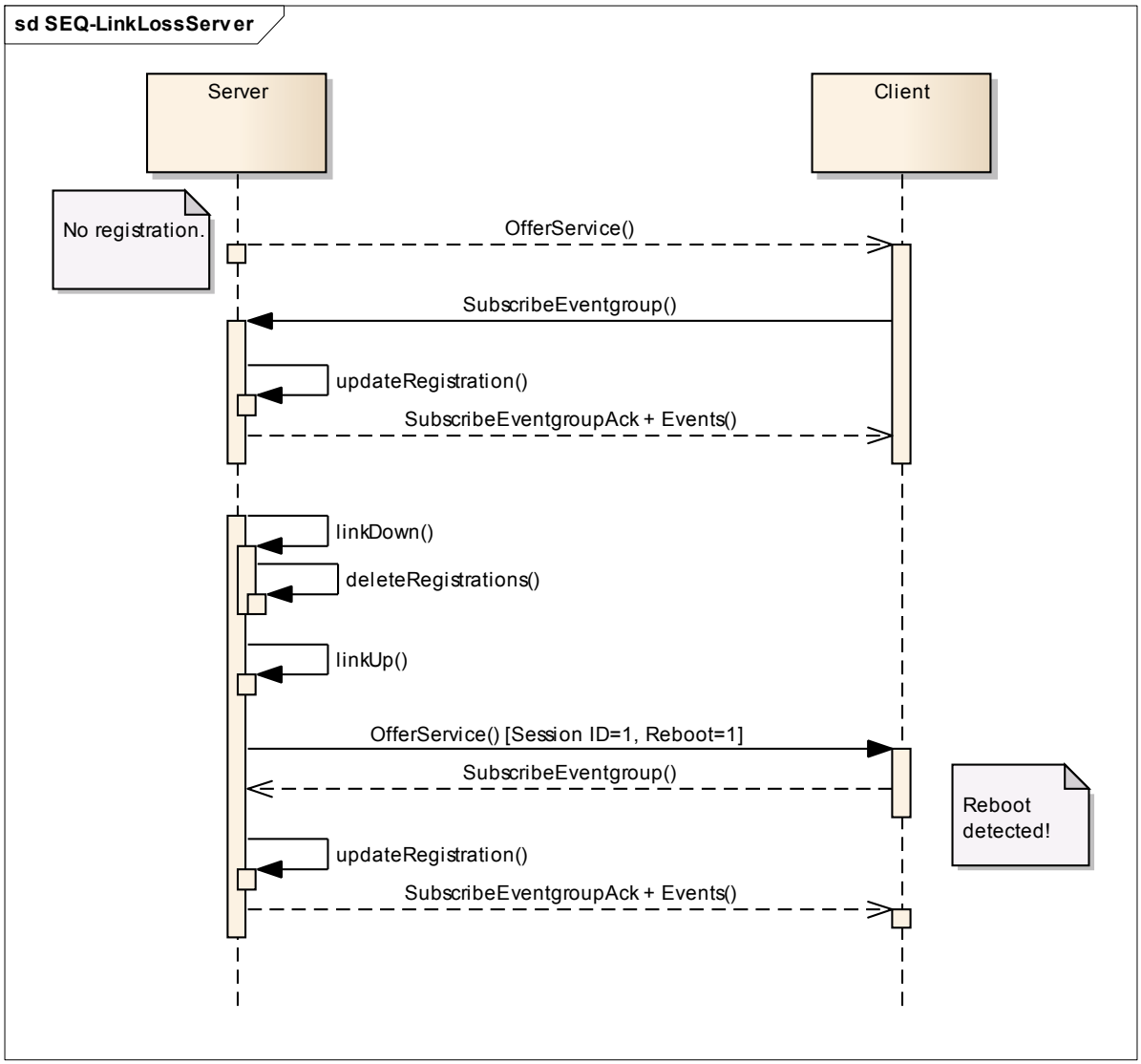

Figure 4.23: Publish/Subscribe with link loss at server (figure ignoring timings)

[PRS_SOMEIPSD_00461] ⌈The client shall open a TCP connection to the server before sending the Subscribe Eventgroup entry if the Service is offered over TCP and the client requests an Eventgroup over TCP according to the configuration.⌋*(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00462] ⌈After a client has sent a Subscribe Eventgroup entry the server shall send a Subscribe Eventgroup Ack entry.⌋*(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00463] ⌈The client shall wait for the Subscribe Eventgroup Ack entry acknowledging a Subscribe Eventgroup entry. If this Subscribe Eventgroup Ack entry does not arrive before the next Subscribe Eventgroup entry is sent, the client shall do the following:

- If the "Explicit Initial Data Control Flag of the Server is set to 0, send a Stop Subscribe Eventgroup entry and a Subscribe Eventgroup entry in the same SOME/IP-SD message the Subscribe Eventgroup entry would have been sent with.

- If the "Explicit Initial Data Control Flag of the Server is set to 1, set the Initial Data Requested Flag of the next Subscribe Eventgroup Entry to 1.

⌋(*RS_SOMEIPSD_00015*) **Note:**
This behavior exists to cope with short durations of communication loss, so new Initial Events are triggered to lower the effects of the loss of messages.

[PRS_SOMEIPSD_00577] ⌈The requirement [PRS_SOMEIPSD_00463] shall not be applied to Offer Service entries that are a reaction to Find Service entries. This means that the Subscribe Eventgroup Ack entry of a Subscribe Eventgroup entry that was triggered by a unicast Offer Service entry is not monitored as well as upon a unicast Offer Service entry the Stop Subscribe Eventgroup entry/Subscribe Eventgroup entry is not sent.⌋(*RS_SOMEIPSD_00015*)

Rationale:
If a client sends a Subscribe Eventgroup entry as a reaction to a unicast offer, and a multicast offer arrives immediately after that but before the the Subscribe Eventgroup Ack entry could be sent by the server and received, the client shall not complain (i.e. Stop Subscribe/Subscribe) about a not yet received acknowledgement.

Note:
This behavior exists to cope with short durations of communication loss. The receiver of a Stop Subscribe Eventgroup and Subscribe Eventgroup combination would sent out Initial Events to lower the effects of the loss of messages.

[PRS_SOMEIPSD_00464] ⌈If the initial value is of concern - i.e. for fields - and the client has the Explicit Initial Data Control Flag (in the SOME/IP-SD header) set to 0, the server shall send the first notifications/events (i.e. initial events) immediately after sending the Subscribe Eventgroup Ack.⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00822] ⌈If the server receives a Subscribe Eventgroup entry with the Initial Data Requested Flag set to 0 and the Explicit Initial Data Control Flag (in the SOME/IP-SD header) set to 1, the server shall send no notifications/events (i.e. initial events).⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00824] ⌈If the server receives a Subscribe Eventgroup entry with the Initial Data Requested Flag set to 1 and the Explicit Initial Data Control Flag (in the SOME/IP-SD header) set to 1, the server shall send notifications/events (i.e. initial events) immediately after sending the Subscribe Eventgroup Ack.⌋(*RS_SOMEIPSD_-00015*)

[PRS_SOMEIPSD_00465] ⌈It is not allowed to send initial values of events upon subscriptions (pure event and not field).⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00120] ⌈The event messages of field notifiers shall be sent on subscriptions (field and not pure event).⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00121] ⌈If a subscription was already valid and is updated by a Subscribe Eventgroup entry, no initial events shall be sent.⌋(*RS_SOMEIPSD_00015*)

[PRS_SOMEIPSD_00122] ⌈Receiving Stop Subscribe / Subscribe combinations trigger initial events of field notifiers.⌋ *(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00123] ⌈The initial events shall be sent after the Subscribe Eventgroup Ack.⌋ *(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00466] ⌈Publish/Subscribe States (server behavior for unicast eventgroups) are defined as follows:

- `Eventgroup_PubSub (Unicast Eventgroup)`
  - `Service Down`
  - `Service Up`
    - `Not Subscribed`
    - `Subscribed`

Initial entry points of `Eventgroup_PubSub (Unicast Eventgroup)` are inside the following states:

- `Eventgroup_PubSub (Unicast Eventgroup)`
  - `Service Up`

Transitions inside `Eventgroup_PubSub (Unicast Eventgroup)` are defined as follows:

FROM entry point `Eventgroup_PubSub (Unicast Eventgroup)`
TO `Service Down`
WITH `[Service==Down]`

FROM `Service Down`
TO `Service Up`
WITH `ServiceUp`

FROM `Service Up`
TO `Service Down`
WITH `ServiceDown`

FROM entry point `Eventgroup_PubSub (Unicast Eventgroup)`
TO `Service UP`
WITH `[Service==Up]`

FROM entry point `Service Up`
TO `Not Subscribed`

FROM `Not Subscribed`
TO `Subscribed`
WITH `receive(SubscribeEventgroup) /enableEvents() send(SubscribeEventgroupAck)`

FROM `Subscribed`
TO `Subscribed`
WITH `receive(SubscribeEventgroup) /send(SubscribeEventgroupAck)`

FROM `Subscribed`
TO `Not Subscribed`
WITH `receive(StopSubscribeEventgroup) /disableEvents()`

FROM `Subscribed`
TO `Not Subscribed`
WITH `TTL_expired [SubscriptionCounter==1] /disableEvents()`

⌋(*RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00016*) Note: Graphical information of the Publish/Subscribe States (server behavior for unicast eventgroups) is shown in Figure 4.24

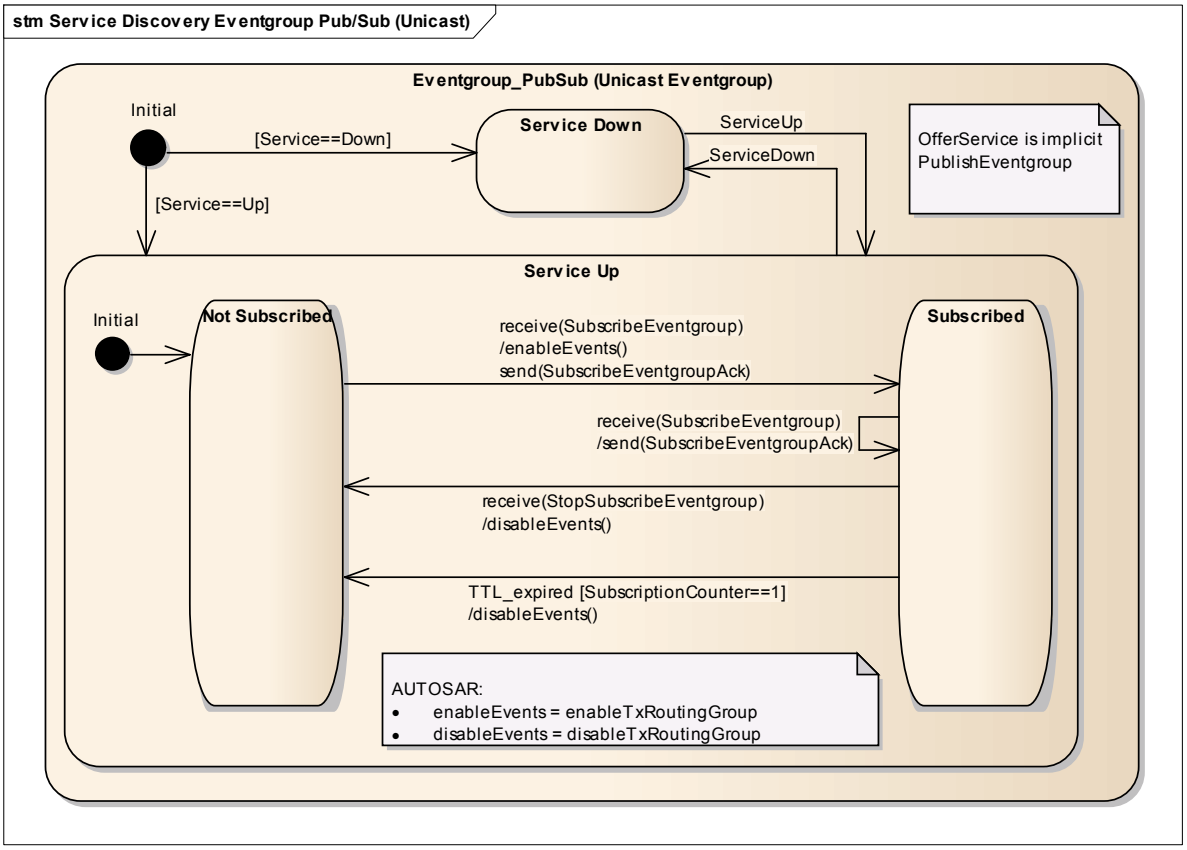

Figure 4.24: Publish/Subscribe State Diagram (server behavior for unicast eventgroups)

[PRS_SOMEIPSD_00571] ⌈If a client subscribes with different SOME/IP-SD messages to different eventgroups of the same Service Instance and all eventgroups in-

clude the same field, the Server shall send out the initial events for this field for every subscription separately.⌋*(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00572] ⌈If a client subscribes with one SOME/IP-SD message to different eventgroups of the same Service Instance and all eventgroups include the same field, the Server may choose to not send out the initial event for this field more than once.⌋*(RS_SOMEIPSD_00015)*

Note:
This means the Server can optimize by sending the initial events only once, if supported by its architecture.

[PRS_SOMEIPSD_00467] ⌈Publish/Subscribe States (server behavior for multicast eventgroups) are defined as follows:

- `Eventgroup_PubSub (Multicast Eventgroup)`
  - `Service Down`
  - `Service Up`
    - `Not Subscribed`
    - `Subscribed`

Initial entry points of `Eventgroup_PubSub (Multicast Eventgroup)` are inside the following states:

- `Eventgroup_PubSub (Multicast Eventgroup)`
  - `Service Up`

Transitions inside `Eventgroup_PubSub (Multicast Eventgroup)` are defined as follows:

FROM entry point `Eventgroup_PubSub (Multicast Eventgroup)`
TO `Service Down`
WITH `[Service==Down]`

FROM `Service Down`
TO `Service Up`
WITH `ServiceUp`

FROM `Service Up`
TO `Service Down`
WITH `ServiceDown`

FROM entry point `Eventgroup_PubSub (Multicast Eventgroup)`
TO `Service UP`

WITH `[Service==Up]`

FROM entry point `Service Up`
TO `Not Subscribed`

FROM `Not Subscribed`
TO `Subscribed`
WITH `receive(SubscribeEventgroup) /enableEvents() SubscriptionCounter++ send(SubscribeEventgroupAck)`

FROM `Subscribed`
TO `Subscribed`
WITH `receive(SubscribeEventgroup) /SubscriptionCounter++ /send(SubscribeEventgroupAck)`

FROM `Subscribed`
TO `Not Subscribed`
WITH `receive(StopSubscribeEventgroup) [SubscriptionCounter==1] /SubscriptionCounter- /disableEvents()`

FROM `Subscribed`
TO `Subscribed`
WITH `receive(StopSubscribeEventgroup) [SubscriptionCounter>1] /SubscriptionCounter-`

FROM `Subscribed`
TO `Not Subscribed`
WITH `TTL_expired [SubscriptionCounter==1] /SubscriptionCounter- disableEvents()`

FROM `Subscribed`
TO `Subscribed`
WITH `TTL_expired [SubscriptionCounter>1] /SubscriptionCounter-`

⌋(*RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00016*) Note: Graphical information of the Publish/Subscribe States (server be-

havior for multicast eventgroups) is shown in Figure 4.25

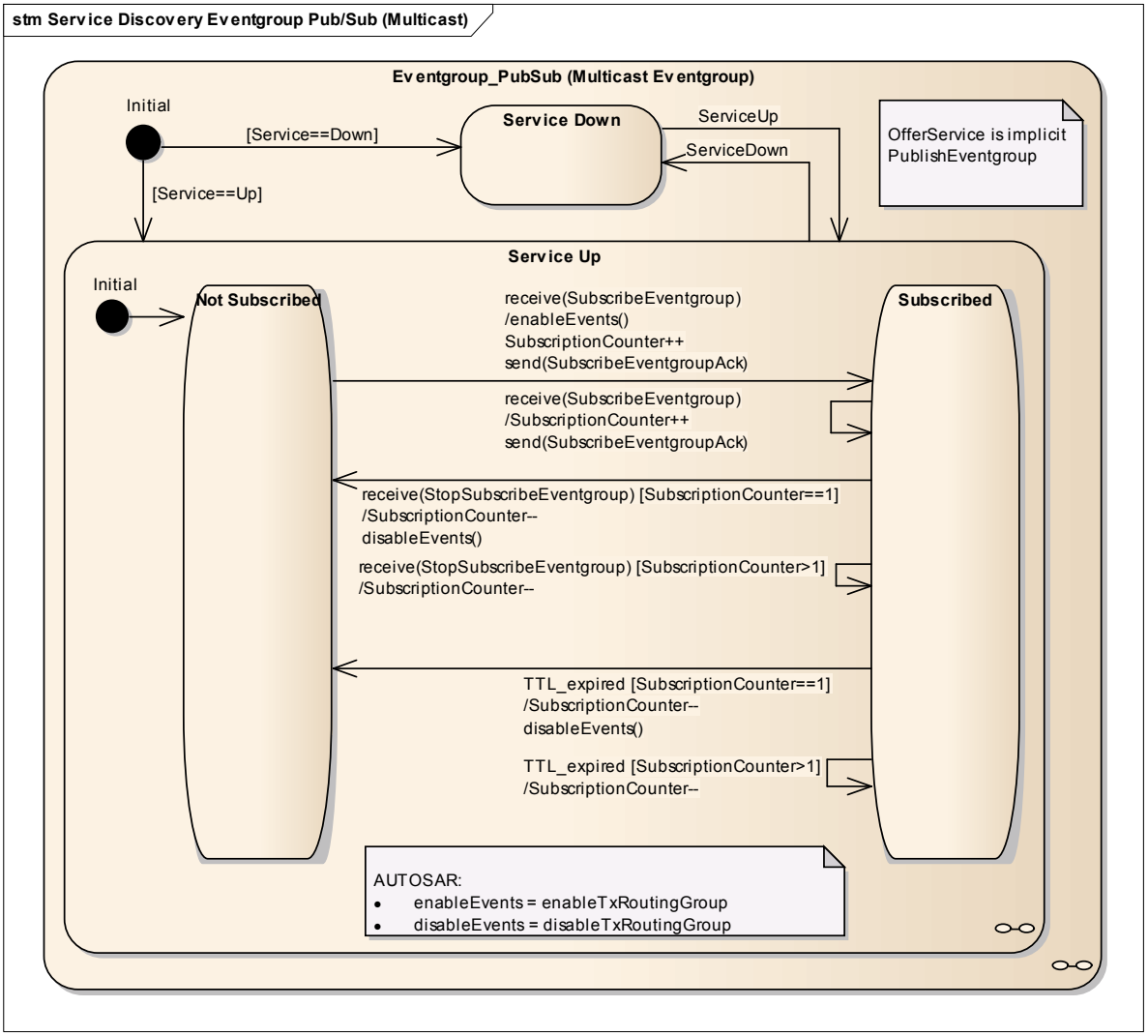

Figure 4.25: Publish/Subscribe State Diagram (server behavior for multicast eventgroups)

[PRS_SOMEIPSD_00468] ⌈Publish/Subscribe States (server behavior for adaptive unicast/multicast eventgroups) are defined as follows:

- Eventgroup_PubSub (Unicast-to-Multicast Eventgroup)
  - Service Down
  - Service Up
    - Not Subscribed
    - Subscribed (Unicast)
    - Subscribed (Multicast)

Initial entry points of `Eventgroup_PubSub (Unicast-to-Multicast Eventgroup)` are inside the following states:

- `Eventgroup_PubSub (Unicast-to-Multicast Eventgroup)`
  - `Service Up`

Transitions inside `Eventgroup_PubSub (Unicast-to-Multicast Eventgroup)` are defined as follows:
FROM entry point `Eventgroup_PubSub (Unicast-to-Multicast Eventgroup)`
TO `Service Down`
WITH `[Service==Down]`

FROM `Service Down`
TO `Service Up`
WITH `ServiceUp`

FROM `Service Up`
TO `Service Down`
WITH `ServiceDown`

FROM entry point `Eventgroup_PubSub (Unicast-to-Multicast Eventgroup)`
TO `Service UP`
WITH `[Service==Up]`

FROM entry point `Service Up`
TO `Not Subscribed`

FROM `Not Subscribed`
TO `Subscribed (Unicast)`
WITH `receive(SubscribeEventgroup) [UnicastLimit>0] /enableEvents() SubscriptionCounter++ send(SubscrieEventgroupAck)`

FROM `Subscribed (Unicast)`
TO `Subscribed (Unicast)`
WITH `receive(SubscribeEventgroup) [UnicastLimit>SubscriptionCounter] /SubscriptionCounter++ send(SubscribeEventgroupAck)`

FROM `Subscribed (Unicast)`
TO `Not Subscribed`
WITH `receive(StopSubscribeEventgroup) [SubscriptionCounter==1] /SubscriptionCounter- disableEvents()`

FROM Subscribed (Unicast)
TO Not Subscribed
WITH TTL_expired [SubscriptionCounter==1] /SubscriptionCounter- disableEvents()

FROM Not Subscribed
TO Subscribed (Multicast)
WITH receive(SubscribeEventgroup) [UnicasLimit==0] /enableMulticastEvents() SubscriptionCounter++ send(SubscribeEventgroupAck)

FROM Subscribed (Multicast)
TO Not Subscribed
WITH receive(StopSubscribeEventgroup) [SubscriptionCounter==1 && UnicasLimit==0] /SubscriptionCounter- disableMulticas- tEvents()

FROM Subscribed (Multicast)
TO Not Subscribed
WITH TTL_expired [SubscriptionCounter==1 && UnicasLimit==0] /SubscriptionCounter- disableMulticastEvents()

FROM Subscribed (Multicast)
TO Subscribed (Multicast)
WITH receive(SubscribeEventgroup) /SubscriptionCounter++

FROM Subscribed (Multicast)
TO Subscribed (Multicast)
WITH receive(StopSubscribeEventgroup) [Subscription- Counter>UnicastLimit+1] /SubscriptionCounter-

FROM Subscribed (Multicast)
TO Subscribed (Multicast)
WITH TTL_expired [SubscriptionCounter>UnicastLimit+1] /SubscriptionCounter-

FROM Subscribed (Unicast)
TO Subscribed (Unicast)
WITH receive(StopSubscribeEventgroup) [SubscriptionCounter>1] /SubscriptionCounter-

FROM `Subscribed (Unicast)`
TO `Subscribed (Unicast)`
WITH `TTL_expired [SubscriptionCounter>1] /SubscriptionCounter-`

FROM `Subscribed (Unicast)`
TO `Subscribed (Multicast)`
WITH `receive(SubscribeEventgroup) [SubscriptionCounter>=UnicastLimit] /SubscriptionCounter++ send(SubscribeEventgroupAck) switchToMulticastEvents()`

FROM `Subscribed (Multicast)`
TO `Subscribed (Unicast)`
WITH `receive(StopSubscribeEventgroup) [SubscriptionCounter==UnicasLimit+1] /switchToUnicastEvents() SubscriptionCounter-`

FROM `Subscribed (Multicast)`
TO `Subscribed (Unicast)`
WITH `TTL_expired [SubscriptionCounter==UnicasLimit+1] /switchToUnicastEvents() SubscriptionCounter-`

⌋(*RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00016*) Note: Graphical information of the Publish/Subscribe States (server behavior

for adaptive unicast/multicast eventgroups) is shown in Figure 4.26

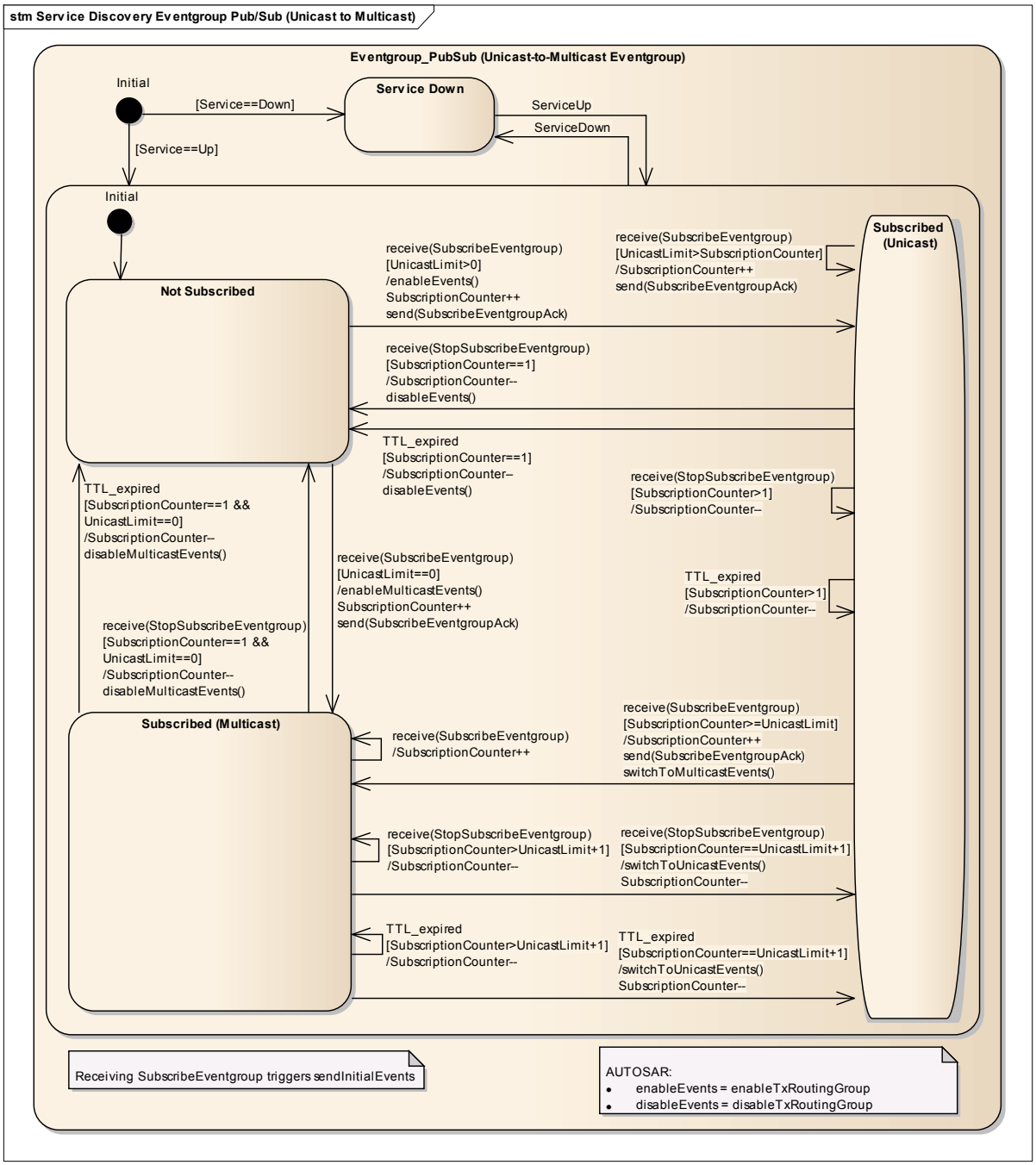

Figure 4.26: Publish/Subscribe State Diagram (server behavior for adaptive unicast/multicast eventgroups)

[PRS_SOMEIPSD_00134] ⌈SOME/IP-SD shall support automated switching from unicast to multicast if a configured threshold of the numbers of subscribers was reached.⌋ *(RS_SOMEIPSD_00025, RS_SOMEIPSD_00016)*

[PRS_SOMEIPSD_00470] ⌈SOME/IP SD Protocol shall support implicit configuration of communication endpoints and registrations of subscribers. These shall be

based on static configurations and not use any SD messages on the network.⌋*(RS_SOMEIPSD_00025, RS_SOMEIPSD_00015)*

Note:
Depending on the project the use case can exist to use services based on a static configuration where no Service Discovery takes place on the network at all. In such cases of implicit registrations, there are no find or subscribe messages exchanged but the services can be used out of the box. Such preconfigurations are not part of SOME/IP or SOME/IP SD. Hence, their configuration and implementation is project specific.

[PRS_SOMEIPSD_00472] ⌈The following entries shall be transported by unicast only:

- Subscribe Eventgroup
- Stop Subscribe Eventgroup
- Subscribe Eventgroup Ack
- Subscribe Eventgroup Nack

⌋*(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00808] ⌈The client shall retry to subscribe to a Eventgroup of a ServerService, if the `SUBSCRIBE_RETRY_MAX` is configured greater than 0. The subscription to the Eventgroup shall be send, if a `SubscribeEventgroupAck/NAck` entry of the requested Eventgroup was not received within a configurable timeout (`SUBSCRIBE_RETRY_DELAY`). The retry shall be done as long as the Eventgroup is requested and the configured retry count (`SUBSCRIBE_RETRY_MAX`) was not exceeded.⌋ *(RS_SOMEIPSD_00015)*

[PRS_SOMEIPSD_00809] ⌈ServerService where the TTL of the received OfferService is set to 0xFFFFFF, could set `SUBSCRIBE_RETRY_MAX` to INF. In this case, the retry shall be done as long as the Eventgroup is requested and no `SubscribeEventgroupAck/NAck` entry of the requested Eventgroup was received.⌋ *(RS_SOMEIPSD_00015)*

4.1.7 Reserved and special identifiers for SOME/IP and SOME/IP-SD.

In this chapter an overview of reserved and special identifiers are shown.

[PRS_SOMEIPSD_00515] ⌈Reserved and special Service-IDs: Table 4.2⌋*(RS_SOMEIPSD_00025)*

Service-ID	Description
0x0000	Reserved
0xFF00 - 0xFF1F	Reserved for Testing at OEM
0xFF20 - 0xFF3F	Reserved for Testing at Tier-1
0xFF40 - 0xFF5F	0xFF5F Reserved for ECU Internal Communication (Tier-1 proprietary)
0xFFFE	Reserved for announcing non-SOME/IP service instances.

0xFFFF	SOME/IP and SOME/IP-SD special service (Magic Cookie, SOME/IP-SD, ...).

Table 4.2: Reserved and Special Service-IDs

[PRS_SOMEIPSD_00516] ⌈Reserved and special Instance-IDs: Table 4.3⌋*(RS_SOMEIPSD_00025)*

Instance-ID	Description
0x0000	Reserved
0xFFFF	All Instances

Table 4.3: Reserved and Special Instance-IDs

[PRS_SOMEIPSD_00517] ⌈Reserved and special Method-IDs/Event-IDs: Table 4.4⌋*(RS_SOMEIPSD_00025)*

Method-ID	Description
0x0000	Reserved
0x7FFF	Reserved
0x8000	Reserved
0xFFFF	Reserved

Table 4.4: Reserved and Special Method/Event-IDs

[PRS_SOMEIPSD_00531] ⌈Reserved eventgroup-IDs: Table 4.5⌋*(RS_SOMEIPSD_00025)*

Eventgroup-ID	Description
0x0000	Reserved
0xFFFF	All Eventgroups

Table 4.5: Reserved Eventgroup-IDs

[PRS_SOMEIPSD_00519] ⌈Method-IDs and Event-IDs of Service 0xFFFF: Table 4.6⌋*(RS_SOMEIPSD_00025)*

Method-ID/Event-ID	Description
0x0000	SOME/IP Magic Cookie Messages
0x8000	SOME/IP Magic Cookie Messages
0x8100	SOME/IP-SD messages (events)

Table 4.6: Method-IDs and Event-IDs of Service 0xFFFF

[PRS_SOMEIPSD_00520] ⌈Besides "otherserv" other names are supported by the configuration option. The following list gives an overview of the reserved names: Table 4.7⌋*(RS_SOMEIPSD_00025)*

Name	Description

hostname	Used to name a host or ECU.
instancename	Used to name an instance of a service.
servicename	Used to name a service.
otherserv	Used for non-SOME/IP Services.

Table 4.7: Reserved Names

5 Configuration Parameters

The Following chapter summarizes all the configuration parameters that are used.

Name	Description
INITIAL_DELAY_MIN	Minimum duration to delay randomly the transmission of a message.
INITIAL_DELAY_MAX	Maximum duration to delay randomly the transmission of a message.
REPETITIONS_BASE_DELAY	Duration of delay for repetitions.
REPETITIONS_MAX	Configuration for the maximum number of repetitions.
REQUEST_RESPONSE_DELAY	The Service Discovery shall delay answers using this configuration item.
CYCLIC_OFFER_DELAY	Interval between cyclic offers in the main phase.
SD_PORT	is a UDP Port for SD Messages (30490 as default).
SD_MULTICAST_IP	address which shall be used by the SD multicast messages.
SUBSCRIBE_RETRY_MAX	Max count of retries for subscribe, as long as the Eventgroup is requested (0=no retry, INF= retry forever (as long as the Eventgroup is requested and no no SubscribeEventgroupAck/Nack entry was received)).
SUBSCRIBE_RETRY_DELAY	Duration of delay to send a consecutive subscribe entries, if a Eventgroup is requested and no SubscribeEventgroupAck/Nack entry was received.

Table 5.1: Configuration Parameters

6 Protocol Usage

6.1 Security Considerations for SOME/IP-SD Options

[PRS_SOMEIPSD_00656] ⌈Received SOME/IP-SD messages shall be checked that the IP Addresses received in Endpoint options and SD Endpoint options are topological correct (reference IP Addresses in the IP subnet for which SOME/IP-SD is used) and shall ignore IP Addresses that are not topological correct as well as the entries referencing those options.⌋*(RS_SOMEIPSD_00025)*

Note:
This means that only Clients and Servers in the same subset are accesible.

6.2 Mandatory Feature Set and Basic Behavior

In this section the mandatory feature set of the Service Discovery and the relevant configuration constraints are discussed. This allow for bare minimum implementations without optional or informational features that might not be required for current use cases.

The following information is defined as compliance check list(s). If a feature is not implemented, the implementation is considered not to comply to SOME/IP-SDs basic feature set.

[PRS_SOMEIPSD_00496] ⌈The following entry types shall be implemented:

- Find Service
- Offer Service
- Stop Offer Service
- Subscribe Eventgroup
- Stop Subscribe Eventgroup
- Subscribe Eventgroup Ack
- Subscribe Eventgroup Nack

⌋*(RS_SOMEIPSD_00008, RS_SOMEIPSD_00013, RS_SOMEIPSD_00014, RS_SOMEIPSD_00015, RS_SOMEIPSD_00017, RS_SOMEIPSD_00007)*

[PRS_SOMEIPSD_00497] ⌈The following option types shall be implemented, when IPv4 is required:

- IPv4 Endpoint Option
- IPv4 Multicast Option
- Configuration Option

- IPv4 SD Endpoint Option (receiving at least)

⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00498] ⌈The following option types shall be implemented, if IPv6 is required:

- IPv6 Endpoint Option
- IPv6 Multicast Option
- Configuration Option
- IPv6 SD Endpoint Option (receiving at least)

⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00500] ⌈The following behaviors/reactions shall be implemented on the Server side:

- The Server shall offer services including the Initial Wait Phase, the Repetition Phase, and the Main Phase depending on the configuration.
- The Server shall offer services using Multicast (Repetition Phase and Main Phase) on the multicast address defined by SD_MULTICAST_IP.
- The Server does not need to answer a Find Service in the Repetition Phase.
- The Server shall answer a Find Service in the Main Phase with an Offer Service using Unicast (no optimization based on unicast flag).
- The Server shall send a Stop Offer Service when shutting down.
- The Server shall receive a Subscribe Eventgroup as well as a Stop Subscribe Eventgroup and react according to this specification.
- The Server shall send a Subscribe Eventgroup Ack and Subscribe Eventgroup Nack using unicast.
- The Server shall support controlling the sending (i.e. fan out) of SOME/IP event messages based on the subscriptions of SOME/IP-SD. This might include sending events based on Multicast.
- The Server shall support the triggering of initial SOME/IP event messages.

⌋(*RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00008*, *RS_SOMEIPSD_00014*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00017*, *RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00501] ⌈The following behaviors/reactions shall be implemented on the Client side:

- The Client shall find services using a Find Service entry and Multicast (on the multicast address defined by SD_MULTICAST_IP) only in the repetition phase.
- The Client shall stop finding a service if the regular Offer Service arrives.

- The Client shall react to the Servers Offer Service with a unicast SD message that includes all Subscribe Eventgroups of the services offered in the message of the Server that the client currently wants to subscribe to.
- The Client shall interpret and react to the Subscribe Eventgroup Ack and Subscribe Eventgroup Nack as specified in this document.

⌋(*RS_SOMEIPSD_00008*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00502] ⌈The following behavior and configuration constraints shall be supported by the Client:

- The Client shall be able handle Eventgroups if only the TTL of the SD Timings is specified. This means that of all the timings for the Initial Wait Phase, the Repetition Phase, and the Main Phase only TTL is configured. This means the client shall only react on the Offer Service by the Server.
- The Client shall answer to an Offer Service with a Subscribe Eventgroup even without configuration of the Request-Response-Delay, meaning it should not wait but answer instantaneously.

⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00020*, *RS_SOMEIPSD_00024*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00503] ⌈The Client and Server shall implement the Reboot Detection as specified in this document and react accordingly. This includes but is not limited to:

- Setting Session ID and Reboot Flag according to this specification.
- Keeping a Session ID counter only used for sending Multicast SD messages.
- Keeping Session ID counters for every Unicast relation for sending Unicast SD messages.
- Understanding Session ID and Reboot Flag according to this specification.
- Keeping a Multicast Session ID counter per ECU that exchanges Multicast SD messages with this ECU.
- Keeping a Unicast Session ID counter per ECU that exchanges Unicast SD messages with this ECU.
- Detecting reboot based on this specification and reaction accordingly.
- Correctly interpreting the IPv4 and IPv6 SD Endpoint Options in regard to Reboot Detection.

⌋(*RS_SOMEIPSD_00018*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00504] ⌈The Client and Server shall implement the "Endpoint Handling for Service and Events". This includes but is not limited to:

- Adding 1 Endpoint Option UDP to an Offer Service if UDP is needed.
- Adding 1 Endpoint Option TCP to an Offer Service if TCP is needed.
- Adding 1 Endpoint Option UDP to Subscribe Eventgroup if events over UDP are required.
- Adding 1 Endpoint Option TCP to Subscribe Eventgroup if events over TCP are required.
- Adding 1 Multicast Option UDP to Subscribe Eventgroup Ack if multicast events are required.
- Understanding and acting according to the Endpoint and Multicast Options transported as described above.
- Overwriting preconfigured values (e.g. IP Addresses and Ports) with the information of these Endpoint and Multicast Options.
- Interpreting incoming IPv4 and IPv6 Endpoint Options as SD endpoints instead of the Address and Port number in the outer layers.

⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00007*)

[PRS_SOMEIPSD_00821] ⌈The Client and Server shall implement the explicit requesting of Initial Events.⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00007*)

6.3 Migration and Compatibility

6.3.1 Supporting multiple versions of the same service.

In order to support migrations scenarios ECUs shall support serving as well as using different incompatible versions of the same service.

[PRS_SOMEIPSD_00512] ⌈In order to support a Service with more than one version the following is required:

- The server shall offer the service instance of this service once per major version.
- The client shall find the service instances once per supported major version or shall use the Major Version as 0xFF (all versions).
- The client shall subscribe to events of the service version it needs.
- All SOME/IP-SD entries shall use the same Service-IDs and Instance-IDs but different Major Versions.

- The server has to demultiplex messages based on the socket they arrive, Message-ID, Major Versions and relay it based on these conditions internally to the correct receiver.

⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00008*, *RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00005*)

[PRS_SOMEIPSD_00806] ⌈In one VLAN there shall be at most one service instance with the same Service ID, Major Version and Instance ID. This applies to the server and to the client Network Nodes.⌋(*RS_SOMEIPSD_00025*, *RS_SOMEIPSD_00008*, *RS_SOMEIPSD_00013*, *RS_SOMEIPSD_00015*, *RS_SOMEIPSD_00005*)

Note: Configuring more than one service instance on one Network Node that differs only in the Minor Version is not allowed since they can not be distinguished in Eventgroup entries.

7 References

References

[1] Glossary
AUTOSAR_TR_Glossary

[2] SOME/IP Protocol Specification
AUTOSAR_PRS_SOMEIPProtocol