Document Title	Specification on SOME/IP Transport Protocol
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	809

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.4.0

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2018-10-31	4.4.0	AUTOSAR Release Management	• Minor corrections • Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Clarification of timeout to monitor successful reception • Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 5

2 Acronyms and abbreviations 6

3 Related documentation 7

3.1 Input documents 7
3.2 Related standards and norms 7
3.3 Related specification 7

4 Constraints and assumptions 8

4.1 Limitations 8
4.2 Applicability to car domains 8

5 Dependencies to other modules 9

5.1 AUTOSAR PDU Router 9
5.2 AUTOSAR Default Error Tracer 9

6 Requirements traceability 10

7 Functional specification 12

7.1 Overview of the SOME/IP header 13
7.1.1 Message Type Field 13
7.1.2 Offset Field 14
7.1.3 Reserved Field 14
7.1.4 More Segments Flag 14
7.1.5 Example 15
7.2 Segmentation of SOME/IP messages (TX Path) 17
7.2.1 Size of SOME/IP segments 17
7.2.2 Header of SOME/IP segments 19
7.2.3 Sending of SOME/IP segments 21
7.2.4 Interruption of the disassembly process 23
7.3 Assembly of received SOME/IP messages (RX path) 24
7.3.1 SOME/IP segment received with Offset 0 26
7.3.2 SOME/IP segment received with Offset> 0 28
7.3.3 Interruption of the assembly process 29
7.4 Error classification 32
7.4.1 Development Errors 32
7.4.2 Runtime Errors 32
7.4.3 Transient Faults 32
7.4.4 Production Errors 32
7.4.5 Extended Production Errors 32

8 API specification 33

8.1 Imported types 33
8.2 Type definitions 33
8.3 Function definitions 34
8.3.1 SomeIpTp_GetVersionInfo 34
8.3.2 SomeIpTp_Init 35

8.3.3 SomeIpTp_Transmit ... 35
8.4 Call-back notifications ... 37
8.4.1 SomeIpTp_TriggerTransmit ... 37
8.4.2 SomeIpTp_RxIndication ... 38
8.4.3 SomeIpTp_TxConfirmation ... 38
8.5 Scheduled functions ... 40
8.5.1 SomeIpTp_MainFunctionTx ... 40
8.5.2 SomeIpTp_MainFunctionRx ... 40
8.6 Expected Interfaces ... 41
8.6.1 Mandatory Interfaces ... 41
8.6.2 Optional Interfaces ... 41
8.6.3 Configurable interfaces ... 42
9 Sequence diagrams ... 43
9.1 Reception ... 43
9.2 Transmission ... 45
10 Configuration specification ... 47
10.1 Containers and configuration parameters ... 47
10.1.1 SomeIpTp ... 47
10.1.2 SomeIpTpGeneral ... 48
10.1.3 SomeIpTpChannel ... 50
10.1.4 SomeIpTpRxNSdu ... 51
10.1.5 SomeIpTpRxNPdu ... 52
10.1.6 SomeIpTpTxNSdu ... 53
10.1.7 SomeIpTpTxNPdu ... 53
11 Not applicable requirements ... 55

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module SOME/IP TP.

The task of the SOME/IP TP module is to segment SOME/IP packets, which do not fit into one single UDP packet. On the reception side, it re-assembles the received SOME/IP segments.

2 Acronyms and abbreviations

Abbreviation / Acronym:	***Description:***
SOME/IP	Scalable service-Oriented MiddlewarE over IP

3 Related documentation

3.1 Input documents

[1] AUTOSAR Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[3] AUTOSAR General Specification for Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

[4] AUTOSAR Requirements on SOME/IP Protocol
AUTOSAR_RS_SOMEIPProtocol.pdf

[5] AUTOSAR SOME/IP Protocol Specification
AUTOSAR_PRS_SOMEIPProtocol.pdf

[6] AUTOSAR PDU Router
AUTOSAR_SWS_PDURouter.pdf

3.2 Related standards and norms

[7] IEC 7498-1 The Basic Model, IEC Norm, 1994

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software (SWS BSW General)[3] which is also valid for the SOME/IP TP module.

Thus, the specification SWS BSW General [3]shall be considered as additional and required specification for SOME/IP TP module.

4 Constraints and assumptions

4.1 Limitations

The SOME/IP TP is a simple protocol to segment SOME/IP messages. It does not implement retry mechanism nor does it reordering of received SOME/IP segments.

These limitations are intended to spare runtime and memory resources on receiver side. Nonetheless, this is a deviation from the AUTOSAR SOME/IP Protocol Specification (PRS_SOMEIP_00747 to PRS_SOMEIP_00754).

The rational for these limitations is the typical use-case which is “streaming” of large SOME/IP messages.

4.2 Applicability to car domains

This module is applicable for SOME/IP communication.

5 Dependencies to other modules

5.1 AUTOSAR PDU Router

The SOME/IP TP module uses the PduR for both directions, the transmission path, and the reception path.

5.2 AUTOSAR Default Error Tracer

In order to be able to report development errors, the SOME/IP TP module has to have access to the error hook of the Default Error Tracer.

6 Requirements traceability

Requirement	Description	Satisfied by
RS_SOMEIP_00010	SOME/IP protocol shall support different transport protocols underneath	SWS_SomeIpTp_00001, SWS_SomeIpTp_00002, SWS_SomeIpTp_00004, SWS_SomeIpTp_00005, SWS_SomeIpTp_00006, SWS_SomeIpTp_00008, SWS_SomeIpTp_00010, SWS_SomeIpTp_00011, SWS_SomeIpTp_00012, SWS_SomeIpTp_00013, SWS_SomeIpTp_00014, SWS_SomeIpTp_00015, SWS_SomeIpTp_00016, SWS_SomeIpTp_00017, SWS_SomeIpTp_00018, SWS_SomeIpTp_00019, SWS_SomeIpTp_00020, SWS_SomeIpTp_00021, SWS_SomeIpTp_00022, SWS_SomeIpTp_00023, SWS_SomeIpTp_00024, SWS_SomeIpTp_00025, SWS_SomeIpTp_00026, SWS_SomeIpTp_00027, SWS_SomeIpTp_00028, SWS_SomeIpTp_00029, SWS_SomeIpTp_00032, SWS_SomeIpTp_00033, SWS_SomeIpTp_00034, SWS_SomeIpTp_00035, SWS_SomeIpTp_00036, SWS_SomeIpTp_00037, SWS_SomeIpTp_00038, SWS_SomeIpTp_00039, SWS_SomeIpTp_00040, SWS_SomeIpTp_00041, SWS_SomeIpTp_00042, SWS_SomeIpTp_00045, SWS_SomeIpTp_00048, SWS_SomeIpTp_00049, SWS_SomeIpTp_00050, SWS_SomeIpTp_00051, SWS_SomeIpTp_00054, SWS_SomeIpTp_00062, SWS_SomeIpTp_00063, SWS_SomeIpTp_00064
RS_SOMEIP_00011	SOME/IP protocol shall support messages of different lengths	SWS_SomeIpTp_00001, SWS_SomeIpTp_00002, SWS_SomeIpTp_00003, SWS_SomeIpTp_00004, SWS_SomeIpTp_00005, SWS_SomeIpTp_00006
RS_SOMEIP_00027	SOME/IP protocol shall define the header layout of messages	SWS_SomeIpTp_00006, SWS_SomeIpTp_00009, SWS_SomeIpTp_00010, SWS_SomeIpTp_00011, SWS_SomeIpTp_00012, SWS_SomeIpTp_00013, SWS_SomeIpTp_00014, SWS_SomeIpTp_00015, SWS_SomeIpTp_00026
RS_SOMEIP_00051	SOME/IP protocol shall provide support for segmented transmission of large data	SWS_SomeIpTp_00002, SWS_SomeIpTp_00004, SWS_SomeIpTp_00005, SWS_SomeIpTp_00009, SWS_SomeIpTp_00012, SWS_SomeIpTp_00035, SWS_SomeIpTp_00042, SWS_SomeIpTp_00048, SWS_SomeIpTp_00063, SWS_SomeIpTp_00064
RS_SOMEIP_00738	-	SWS_SomeIpTp_00019, SWS_SomeIpTp_00023, SWS_SomeIpTp_00024, SWS_SomeIpTp_00025, SWS_SomeIpTp_00041, SWS_SomeIpTp_00050, SWS_SomeIpTp_00051
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_SomeIpTp_00043
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_SomeIpTp_00058, SWS_SomeIpTp_00069
SRS_BSW_00384	The Basic Software	SWS_SomeIpTp_00060, SWS_SomeIpTp_00061

	Module specifications shall specify at least in the description which other modules they require	
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_SomeIpTp_00044, SWS_SomeIpTp_00046
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_SomeIpTp_00044, SWS_SomeIpTp_00046
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_SomeIpTp_00058, SWS_SomeIpTp_00059, SWS_SomeIpTp_00069, SWS_SomeIpTp_00070

7 Functional specification

The task of the SOME/IP TP module is to segment SOME/IP packets, which do not fit into one single UDP packet. On the reception side, it assembles the received SOME/IP segments.

The SOME/IP TP module interacts with the PDU Router for both directions, the transmission and the reception path.

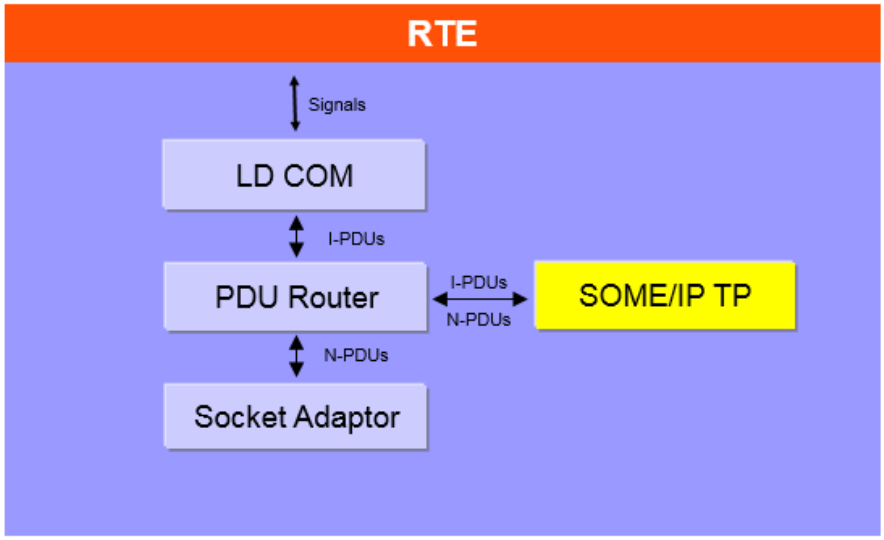


Figure 2 – Location of the SOME/IP TP module

7.1 Overview of the SOME/IP header

This chapter describe the relevant parts of the SOME/IP header for the segmentation of SOME/IP messages.

The Message Type field of the SOME/IP header contains a bit, which marks the SOME/IP PDU as a segment of an original SOME/IP message. Every segmented SOME/IP message adds SOME/IP TP specific fields to the SOME/IP header.

These fields contain control information for the segmentation and the reassembly of original, large SOME/IP messages. How they are used is described in the following chapters.

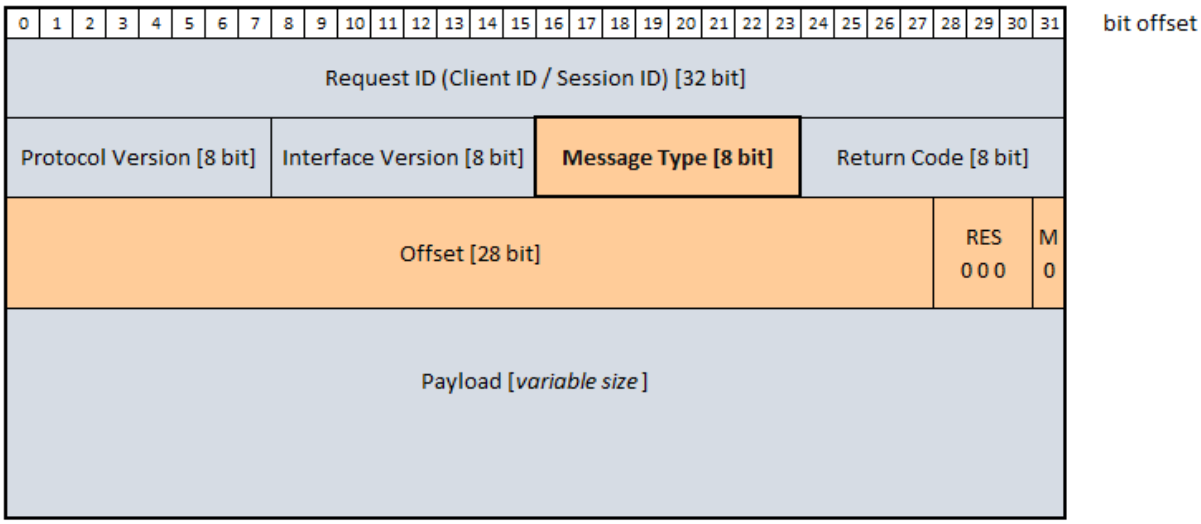


Figure 3 –SOME/IP TP header

Note: The Offset Field, the Reserved bits and the More Segment Flag are only present if the TP-Flag is set to ‘1’.

7.1.1 Message Type Field

The Message Type Field contains the TP-Flag, which marks this SOME/IP message as a SOME/IP segment of an original SOME/IP message.

	Message Type [8 bit]							
bit offset	16	17	18	19	20	21	22	23
Value	x	x	**0/1**	x	x	x	x	x
Name	*ignore*	*ignore*	TP-Flag	*ignore*	*ignore*	*ignore*	*ignore*	*ignore*

Figure 4 – Location of the TP-Flag

7.1.2 Offset Field

The Offset Field [28 bits] is located right after the Return Code field. It starts at bit offset 0, and ends at bit offset 27. The contained value increases after every transmitted/received segment according to the payload length of the previous transmitted/received SOME/IP segment.

The **Offset Field** contains the **Offset Value** in units of 16 bytes. (E.g.: If the Offset Field is set to 92, 1472 Payload bytes have been transmitted so far.) These two different terms are used in the remainder of this document.

Note: The payload length provided in the Offset Field does not include the bytes which are needed for the SOME/IP header.

7.1.3 Reserved Field

The Reserved Field [3 bits] follows the Offset Field. It starts at bit offset 28 and ends at bit offset 30. These three bits are reserved and set to 0.

7.1.4 More Segments Flag

The More Segments Flag [1 bit] indicates whether another segmented SOME/IP PDU will follow.

7.1.5 Example

An original SOME/IP message of 5571 bytes payload has to be transmitted.
The Length field of this original SOME/IP message is set to 8 + 5571 bytes.

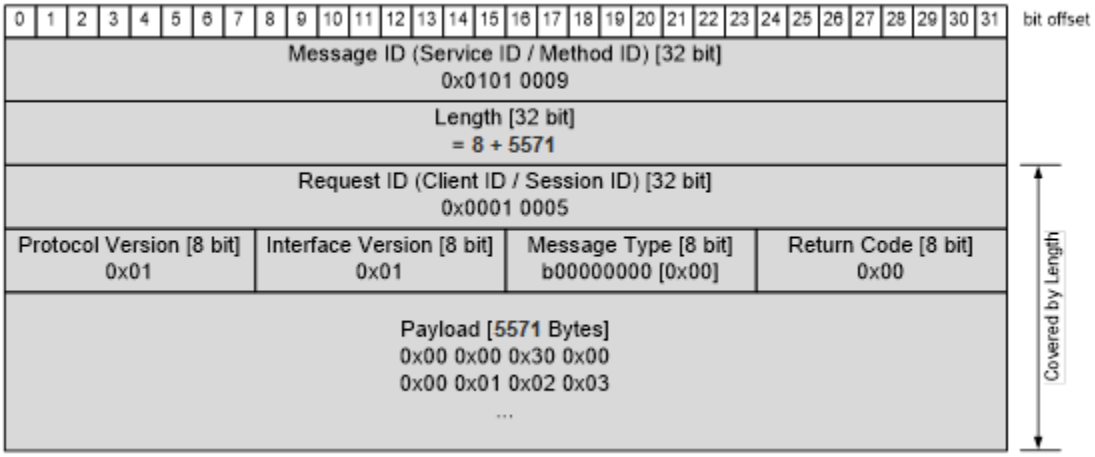


Figure 5 – Example: Header of Original SOME/IP message

This original SOME/IP message will now be segmented into 5 consecutive SOME/IP segments. Every payload of these segments carries at most 1392 bytes in this example.

For these segments, the SOME/IP TP module adds additional TP fields (marked red). The Length field of the SOME/IP carries the overall length of the SOME/IP segment including 8 bytes for the Request ID, Protocol Version, Interface Version, Message Type and Return Code. Because of the added TP fields (4 bytes), this Length information is extended by 4 additional SOME/IP TP bytes.

The following figure provides an overview of the relevant SOME/IP header settings for every SOME/IP segment:

	Length (Bytes)	Message Type [TP-Flag]	Offset Value	More Segment Flag
1st segment	8 + 4 + 1392 = 1404	TP-Flag = ‘1’	0	1
2nd segment	8 + 4 + 1392 = 1404	TP-Flag = ‘1’	87	1
3rd segment	8 + 4 + 1392 = 1404	TP-Flag = ‘1’	174	1
4th segment	8 + 4 + 1392 = 1404	TP-Flag = ‘1’	261	1
5th segment	8 + 4 + 312 = 324	TP-Flag = ‘1’	348	0

Figure 6 – Example: Overview of relevant SOME/IP TP headers

Note: Please be aware that the value provided within the Offset Field is given in units of 16 bytes, i.e.: The Offset Value of 87 correspond to 1392 bytes Payload.

The complete SOME/IP headers of the SOME/IP segments message will look like this in detail:

- The first 4 segments contain 1392 Payload bytes each with “More Segments Flag” set to ‘1’:

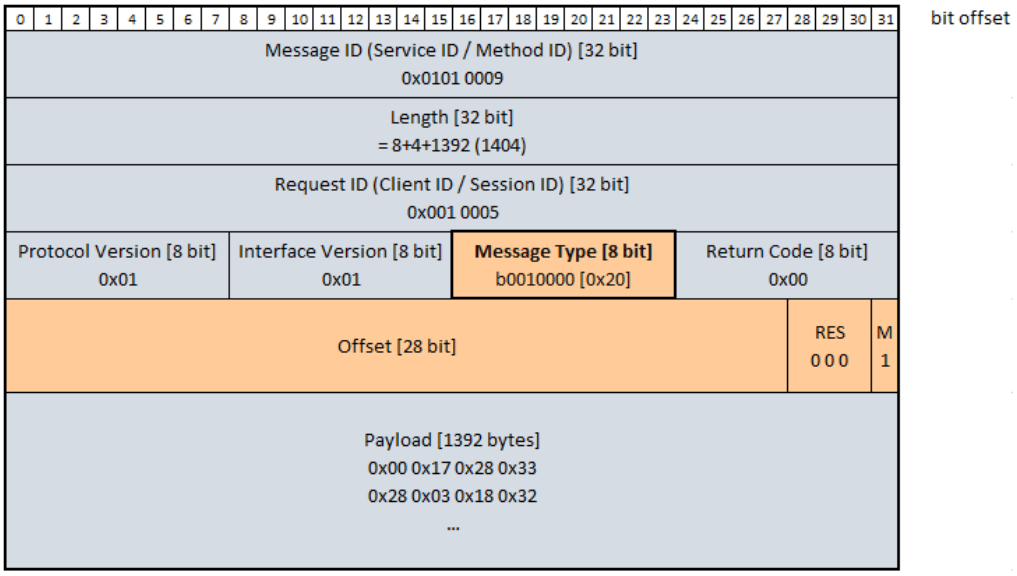


Figure 7 – Example: Header of the SOME/IP segments

- The last segment (i.e. #5) contains the remaining 312 Payload bytes of the original 5880 bytes payload. This last segment is marked with “More Segments flag” set to ‘0’.

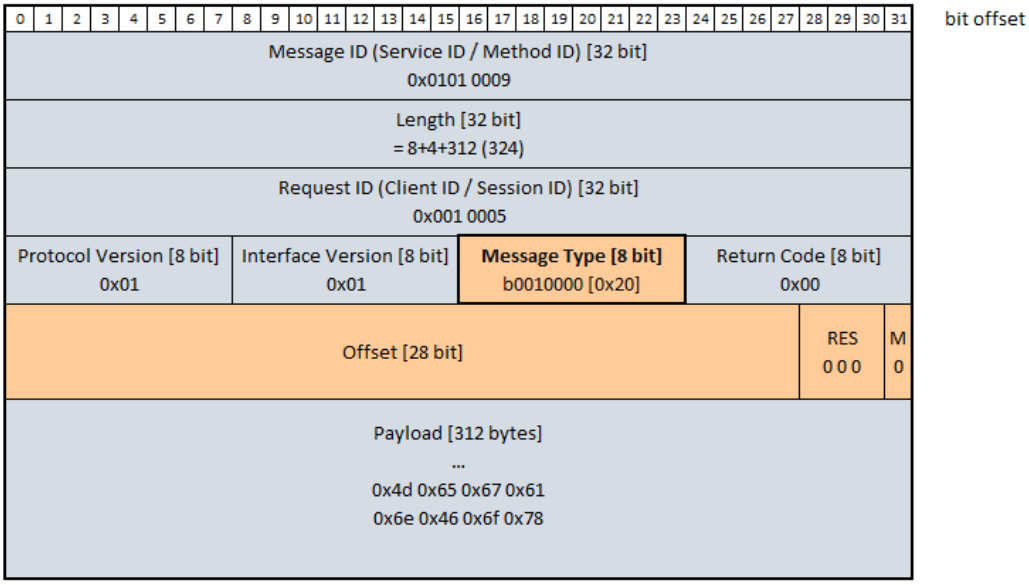


Figure 8 – Example: Header of the last SOME/IP segment

7.2 Segmentation of SOME/IP messages (TX Path)

The following chapter describe the necessary activities of the SOME/IP TP module to segment SOME/IP messages.

7.2.1 Size of SOME/IP segments

[SWS_SomeIpTp_00001]⌈
The SOME/IP TP module shall remember the PDU length separately for every PDU ID which is passed by the `PduInfoPtr` parameter of the `SomeIpTp_Transmit()` call.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00011)

Note:
The SOME/IP TP module needs this information to calculate the payload size, the Offset Value, and the More Segments Flag for the SOME/IP segments which are going to be transmitted.

[SWS_SomeIpTp_00002]⌈
The amount of generated SOME/IP segments shall be as little as possible.
⌋ (RS_SOMEIP_00011, RS_SOMEIP_00010, RS_SOMEIP_00051)

Note: This means that the SOME/IP TP module shall try to always use the maximum allowed segmentation size.

[SWS_SomeIpTp_00003]⌈
The size of every segmented SOME/IP message shall consist of the sum of 12 bytes of SOME/IP header, and the Payload bytes itself.
⌋ (RS_SOMEIP_00011)

[SWS_SomeIpTp_00004]⌈
The SOME/IP TP module shall derive the maximum possible size of the segmented SOME/IP PDUs using the parameter `SomeIpTpTxNPduRef`.
⌋ (RS_SOMEIP_00011, RS_SOMEIP_00010, RS_SOMEIP_00051)

[SWS_SomeIpTp_00005]⌈
The SOME/IP TP module shall generate segmented SOME/IP PDUs not larger than the size derived from the parameter `SomeIpTpTxNPduRef`.
⌋ (RS_SOMEIP_00011, RS_SOMEIP_00010, RS_SOMEIP_00051)

[SWS_SomeIpTp_00006]⌈
Every payload of a segmented SOME/IP message except the last one has to be a multiple of 16 bytes.
⌋ (RS_SOMEIP_00011, RS_SOMEIP_00010, RS_SOMEIP_00027)

Note:
The last segment may consist of an odd payload or a payload which is not dividable by 16. The amount of the contained payload bytes are written into the Length field of the SOME/IP header.

[SWS_SomeIpTp_00007]⌈
The SOME/IP TP module shall buffer the pointer to the Meta-data for every PDU ID separately which is passed by the PduInfoPtr parameter of the API `SomeIpTp_Transmit()`, and forward this information when `PduR_SomeIpTpTransmit()` is called for each segment.
⌋ ()

7.2.2 Header of SOME/IP segments

Every generated SOME/IP header for each SOME/IP segment is set to the following values:

The following fields are received by the upper layer:

- Request ID [32 bit] –derived value, see SWS_SomeIpTp_00007
- Protocol Version [8 bit] – derived value, see SWS_SomeIpTp_00007
- Interface Version [8 bit] – derived value, see SWS_SomeIpTp_00007
- Message Type [8 bit] – calculated value, see SWS_SomeIpTp_00008
- Return Code [8 bit] – derived value, see SWS_SomeIpTp_00007

The following fields are added by the SOME/IP TP module:

- Offset [28 bit] – calculated value, see SWS_SomeIpTp_00011
- Reserved bits [3 bit] – statically set to '000', see SWS_SomeIpTp_00012
- More Segment Flag [1 bit] – calculated value, see SWS_SomeIpTp_00013

[SWS_SomeIpTp_00008]⌈
The SOME/IP TP module shall store the Request ID, Protocol Version, Interface Version, Message Type, and the Return Code of the SOME/IP header for every PDU ID separately which is returned by the first call of `PduR_SomeIpTpCopyTxData()` triggered by the API call `SomeIpTp_Transmit()`.
⌋ (RS_SOMEIP_00010)

Note:
The SOME/IP header is contained in the first 8 bytes of the total length of the original SOME/IP PDU. The total length is provided via the API call `SomeIpTp_Transmit()`.

[SWS_SomeIpTp_00009]⌈
If the provided SDU fits into one single PDU, the provided SOME/IP header shall be used with no modification.

If the provided SDU does not fit into one single SOME/IP PDU, the SOME/IP TP module shall set the TP-Flag of the Message Type to '1' for every SOME/IP segment which is going to be sent on the bus via the PduR.
All the other bits contained in the Message Type field shall stay untouched.

⌋ (RS_SOMEIP_00027, RS_SOMEIP_00051)

[SWS_SomeIpTp_00010]⌈
The SOME/IP TP module shall create and attach the Offset Field, the Reserved bits, and the More Segment Flag to every SOME/IP segment which is going to be sent on the bus.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00027)

[SWS_SomeIpTp_00011]⌈
The Offset Field of the first SOME/IP segment shall be set to ‘0’.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00027)

[SWS_SomeIpTp_00012]⌈
The SOME/IP TP module shall increase the value of the Offset Field for every successfully transmitted SOME/IP segment by the amount of bytes which have been transmitted by the previous SOME/IP segment divided by 16.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00027, RS_SOMEIP_00051)

[SWS_SomeIpTp_00013]⌈
The SOME/IP TP module shall set the Reserved bits statically to ‘000’ by the sender and shall be ignored by the receiver.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00027)

[SWS_SomeIpTp_00014]⌈
The SOME/IP TP module shall set the More Segment Flag to ‘1’ except for the last SOME/IP segment.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00027)

[SWS_SomeIpTp_00015]⌈
The SOME/IP TP module shall set the More Segment Flag to ‘0’ for the last SOME/IP segment.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00027)

7.2.3 Sending of SOME/IP segments

[SWS_SomeIpTp_00016]⌈
If the API `SomeIpTp_Transmit()` is called, the SOME/IP TP module shall check for an ongoing segmentation for the provided PDU ID.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00017]⌈
If the API `SomeIpTp_Transmit()` is called while no segmentation is ongoing for this PDU ID, the SOME/IP TP module shall perform the following steps in the following order:

- Remember the provided PDU length (provided PduInfoPtr).
- Derive the PDU ID which shall be used for every segmented SOME/IP PDU (see `SomeIpTpTxNPduRef`).
- Calculate the size of the SOME/IP for the first segment (considering header and payload)
- Call the API `PduR_SomeIpTpTransmit()` using the derived PDU ID and the calculated PDU size and set the SduDataPtr to NULL_PTR.

⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00018]⌈
When the API `SomeIpTp_TriggerTransmit()` is called, create the header for the SOME/IP segment and call the API `PduR_SomeIpTpCopyTxData()` using the calculated payload for this segment, and set the parameter `retry` to NULL_PTR.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00019]⌈
To calculate the possible maximum size for all consecutive SOME/IP TP segments, the SOME/IP TP module shall consider the available buffer size of the upper layer by evaluating the `availableDataPtr`, provided by the API `PduR_SomeIpTpCopyTxData()`.
I.e.:The payload size of the next SOME/IP TP segment needs to be smaller or equal to the available buffer, AND dividable by 16 for all segments but for the last.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00738)

[SWS_SomeIpTp_00020]⌈
The SOME/IP TP module shall debounce subsequent calls of the API `PduR_SomeIpTpTransmit()` for the same PDU ID,using the parameter `SomeIpTpNPduSeparationTime`.

It defines the time span between the call of `SomeIpTp_TxConfirmation()`, and the subsequent call of the API `PduR_SomeIpTpTransmit()`.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00021]⌈
If the last SOME/IP segment of the original SOME/IP PDU has been transmitted successfully (i.e. the call of `SomeIpTp_TxConfirmation()` with parameter success equals TRUE occurred for the last call of `PduR_SomeIpTpCopyTxData()`), the SOME/IP TP module shall

- Call the API `PduR_SomeIpTpTxConfirmation()`.

⌋ (RS_SOMEIP_00010)

Note:
With the call of `PduR_SomeIpTpTxConfirmation()`, the segmentation process is finished.

7.2.4 Interruption of the disassembly process

[SWS_SomeIpTp_00022][
If the API `SomeIpTp_Transmit()` is called with a PDU ID which is currently used for an ongoing segmentation,

- E_NOT_OK shall be returned.
- The ongoing disassembly process for this PDU ID shall be canceled.
- The API `PduR_SomeIpTpTxConfirmation()` with result set to E_NOT_OK shall be called.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

] (RS_SOMEIP_00010)

[SWS_SomeIpTp_00023][
If the API `SomeIpTp_TxConfirmation()` is called with parameter success set to FALSE,

- The disassembly process for this PDU ID shall be canceled.
- The API `PduR_SomeIpTpTxConfirmation()` with `result` set to E_NOT_OK shall be called.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

] (RS_SOMEIP_00010, RS_SOMEIP_00738)

[SWS_SomeIpTp_00024][
If the `availableDataPtr`, provided by the API `PduR_SomeIpTpCopyTxData()` is smaller than 16 bytes, or smaller than required for the last SOME/IP TP segment,

- The disassembly process for this PDU ID shall be canceled.
- The API `PduR_SomeIpTpTxConfirmation()` with `result` set to E_NOT_OK shall be called.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

] (RS_SOMEIP_00010, RS_SOMEIP_00738)

[SWS_SomeIpTp_00025][
If an API `PduR_SomeIpTpCopyTxData()` returns something else than BUFREQ_OK,

- The disassembly process for this PDU ID shall be canceled.
- The API `PduR_SomeIpTpTxConfirmation()` with `result` set to E_NOT_OK shall be called.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

] (RS_SOMEIP_00010, RS_SOMEIP_00738)

7.3 Assembly of received SOME/IP messages (RX path)

[SWS_SomeIpTp_00026]⌈
If the API `SomeIpTp_RxIndication()` is called, the SOME/IP TP module shall derive the following SOME/IP header information from the first 12 bytes of the received PDU:

- Request ID [32 bit]
- Protocol Version [8 bit]
- Interface Version [8 bit]
- Message Type [8 bit]
- Return Code [8 bit]
- Offset [28 bit]
- Reserved bits [3 bit]
- More Segment Flag [1 bit]

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00010, RS_SOMEIP_00027)

[SWS_SomeIpTp_00027]⌈
The SOME/IP TP module shall be able to store the value of the Offset Field for every PDU ID separately.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00028]⌈
The SOME/IP TP module shall be able to store the number of Payload bytes for every PDU ID separately which has been passed by a call of `SomeIpTp_RxIndication()`.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00029]⌈
The SOME/IP TP module shall store the status of the More Segment Flag for every PDU ID separately which is passed by a call of `SomeIpTP_RxIndication()`.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00030]⌈
The SOME/IP TP module shall buffer the pointer to the Meta-data for every PDU ID separately which is passed by the PduInfoPtr parameter of the API `SomeIpTp_RxIndication()`, and forward this information when `PduR_SomeIpTpStartOfReception` is called.
⌋ ()

[SWS_SomeIpTp_00031]⌈
If `SomeIpTp_RxIndication()` is called with TP Flag set to ‘0’, SOME/IP TP shall call `PduR_SomeIpTpStartOfReception`, `PduR_SomeIpTpCopyRxData()`, and `PduR_SomeIpTpRxIndication()`, directly after each other providing the received indication.
⌋ ()

[SWS_SomeIpTp_00071]⌈
If `SomeIpTp_RxIndication()` is called with

- TP Flag set to ‘1’,
- Offset Field set to '0', and
- More Segment Flag set to '0',

SOME/IP TP shall call `PduR_SomeIpTpStartOfReception()`, `PduR_SomeIpTpCopyRxData()`, and `SomeIpTp_RxIndication()`, directly after each other providing the received indication.
⌋ ()

7.3.1 SOME/IP segment received with Offset 0

[SWS_SomeIpTp_00032]⌈
If a SOME/IP segment is successfully received with Offset Field set to 0, the SOME/IP TP module shall store the values of the received SOME/IP header for each PDU ID separately.These values shall be used as reference values for the (expected) following consecutive receiving SOME/IP segments (i.e. with Offset Field set to > 0).
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00033]⌈
If a SOME/IP segment is successfully received with Offset Field set to 0, the SOME/IP TP module shall

- Start the Rx timeout time defined by `SomeIpTpRxTimeoutTime`.
- Call the API `PduR_SomeIpTpStartOfReception()` with the PDU ID derived from the parameter `SomeIpTpRxSduRef` and the `TpSduLength` set to '0'.

⌋ (RS_SOMEIP_00010)

Note:
`TpSduLength` set to '0' indicates "unknown message length" to the upper layers.

[SWS_SomeIpTp_00034]⌈
If a SOME/IP segment is successfully received with Offset Field set to 0 and after the SOME/IP TP module has called the API `PduR_SomeIpTpStartOfReception()`, the SOME/IP TP module shall check the returned `bufferSizePtr`.

If the `bufferSizePtr` is greater or equal to the sum of the received payload and the added SOME/IP header, the SOME/IP TP module shall call the API `PduR_SomeIpTpCopyRxData()` to pass the following assembled SOME/IP message. This PDU shall contain the following content:

- Request ID [32 bit]
- Protocol Version [8 bit]
- Interface Version [8 bit]
- Message Type [8 bit] - see SWS_SomeIpTp_00028
- Return Code [8 bit]
- *[Payload]*

⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00035]⌈
The SOME/IP TP module shall set the TP-Flag contained in the Message Type back to '0' before the assembled SOME/IP header is passed to the upper layer.
⌋ (RS_SOMEIP_00010, RS_SOMEIP_00051)

[SWS_SomeIpTp_00036][
The SOME/IP TP module shall store the number of Payload bytes for every PDU ID separately which has been passed to the upper layer.
] (RS_SOMEIP_00010)

Note:
This information will be used to verify the Offset Value of the consecutive SOME/IP segments.

7.3.2 SOME/IP segment received with Offset> 0

[SWS_SomeIpTp_00037]⌈
If a SOME/IP segment is successfully received with Offset Field> 0, the SOME/IP TP module shall compare the received SOME/IP header fields with the values of the stored SOME/IP header fields which has been received with the first segment (i.e. Offset was set to 0):

- Request ID [32 bit]
- Protocol Version [8 bit]
- Interface Version [8 bit]
- Message Type [8 bit]
- Return Code [8 bit]

If these values match restart the `SomeIpTpRxTimeoutTime` and continue with the assembly process.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00038]⌈
The SOME/IP TP module shall store the number of Payload bytes for every PDU ID separately which has been passed to the upper layer.
⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00039]⌈
The SOME/IP TP module shall compare the value of the Offset Field with the sum divided by 16 of copied Payload bytes since the first received SOME/IP segment (i.e. with Offset Field set to ‘0’).
If this sum divided by 16 matches with the current Offset Value and if the `bufferSizePtr` provided by the previous call of the API `PduR_SomeIpTpCopyRxData()` is greater or equal to the received payload, call the API `PduR_SomeIpTpCopyRxData()` with `SduLength` set to the received Payload bytes.
⌋ (RS_SOMEIP_00010)

Note:
In case of Offset Field value > 0, only the Payload bytes are provided to the upper layer (without any SOME/IP header fields)

[SWS_SomeIpTp_00040]⌈
If a SOME/IP segment is successfully received with the More Segment Flag set to ‘0’, the SOME/IP TP module shall

- Cancel the Rx timeout time defined by `SomeIpTpRxTimeoutTime`.
- Call the API `PduR_SomeIpTpRxIndication()` after it has copied the remaining received Payload bytes to the upper layer(as defined in SWS_SomeIpTp_00033).

⌋ (RS_SOMEIP_00010)

7.3.3 Interruption of the assembly process

[SWS_SomeIpTp_00041]⌈
If the Rx timeout time defined by `SomeIpTpRxTimeoutTime` expires,

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00738)

[SWS_SomeIpTp_00042]⌈
If the API `SomeIpTp_RxIndication()` is called with the Offset Value is > 0 but no session is currently running,

- The received PDU shall be ignored
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_INCONSISTENT_SEQUENCE.

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00051)

Note: This check identifies that at least the first segment has not been received.

[SWS_SomeIpTp_00054]⌈
If the SOME/IP TP module interrupts the assembly process because of a detected error, the SOME/IP TP module shall

- Call the API `PduR_SomeIpTpRxIndication()` for this PDU ID with E_NOT_OK.
- The Rx timeout time defined by `SomeIpTpRxTimeoutTime` shall be canceled (if still running) for this PDU ID.

⌋ (RS_SOMEIP_00010)

Note: The possible reasons for interruptions are listed below.

[SWS_SomeIpTp_00062]⌈
If the SOME/IP TP module detects an inconsistency of the received SOME/IP TP headers (i.e.: Request ID, Protocol Version, Interface Version, Message Type or Return Code are not equal for all received segments),

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_INCONSISTENT_HEADER.

⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00045]⌈
If the API `SomeIpTp_RxIndication()` is called and a session is currently active, the SOME/IP TP module shall check if the TP-Flag of the Message Type is set to ‘1’.
If the TP-Flag is not set to ‘1’,

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_MESSAGE_TYPE.

⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00063]⌈
If the API `SomeIpTp_RxIndication()` is called, the SOME/IP TP module shall check whether the received payload bytes are dividable by16 in case the More Segment Flag is set to ‘1’.
If the received payload bytes are not dividable by 16 in this case,

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00051)

[SWS_SomeIpTp_00064]⌈
If the API `SomeIpTp_RxIndication()` is called, the SOME/IP TP module shall check the value of the Offset Field. If the Offset Value in units of 16 bytes does not match to the sum of the received Payload bytes of the previous SOME/IP segments,

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_INCONSISTENT_SEQUENCE.

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00051)

[SWS_SomeIpTp_00048]⌈
If the API `SomeIpTp_RxIndication()` is called, the SOME/IP TP module shall check the value of the Offset Field. If the received Offset Value equals ‘0’ while the received Payload bytes of the previous SOME/IP segments is greater than ‘0’, the SOME/IP TP module shall perform the following steps in the following order:

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_INCONSISTENT_SEQUENCE.
- Start the assembly process according to chapter 7.3.1 SOME/IP segment received with Offset 0

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00051)

[SWS_SomeIpTp_00049]⌈
If the bufferSizePtr provided by the API `PduR_SomeIpTpStartOfReception()` or `PduR_SomeIpTpCopyRxData()` is smaller than the sum of the received and the added SOME/IP header (in case of the first segment) or the received payload (in case of any subsequent segment),

- The current assembly process shall be interrupted as defined by SWS_SomeIpTp_00054.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋ (RS_SOMEIP_00010)

[SWS_SomeIpTp_00050]⌈
If the API `PduR_SomeIpTpCopyRxData()` returns something else than BUFREQ_OK,

- The assembly process for this PDU ID shall be interrupted as defined by SWS_SomeIpTp_00054.
- .
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00738)

[SWS_SomeIpTp_00051]⌈
If the API `PduR_SomeIpTpStartOfReception()` returns something else than BUFREQ_OK,

- The assembly process for this PDU ID shall be stopped.
- The API `Det_ReportRuntimeError()` shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋ (RS_SOMEIP_00010, RS_SOMEIP_00738)

7.4 Error classification

7.4.1 Development Errors

The following Development Errors can be detected by the SOME/IP TP module

[SWS_SomeIpTp_00052][Development Error Types

Type of error	*Related error code*	*Value [hex]*
SOME/IP TP module not initialized	SOMEIPTP_E_UNINIT	0x01
Null pointer has been passed as an argument	SOMEIPTP_E_PARAM_POINTER	0x02
Unknown parameter has been passed	SOMEIPTP_E_PARAM	0x03

] ( )

7.4.2 Runtime Errors

[SWS_SomeIpTp_00065] Runtime Error Types

Type of error	*Related error code*	*Value [hex]*
The TP-Flag (of Message Type) was set to ‚0‘	SOMEIPTP_E_MESSAGE_TYPE	0x04
Inconsistent subsequent segment received	SOMEIPTP_E_INCONSISTENT_SEQUENCE	0x05
Inconsistent header received	SOMEIPTP_E_INCONSISTENT_HEADER	0x06
Disassembly Interrupt due to the upper layer	SOMEIPTP_E_DISASSEMBLY_INTERRUPT	0x07
Assembly Interrupt due to the upper layer	SOMEIPTP_E_ASSEMBLY_INTERRUPT	0x08

] ( )

7.4.3 Transient Faults

There are no extended production errors.

7.4.4 Production Errors

There are no production errors.

7.4.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following modules are listed:

[SWS_SomeIpTp_00043] [

Module	*Header File*	*Imported Type*
ComStack_Types	ComStackTypes.h	BufReq_ReturnType
	ComStackTypes.h	PduIdType
	ComStackTypes.h	PduInfoType
	ComStackTypes.h	PduLengthType
	ComStackTypes.h	RetryInfoType
Std_Types	StandardTypes.h	Std_ReturnType
	StandardTypes.h	Std_VersionInfoType

] (SRS_BSW_00301)

8.2 Type definitions

[SWS_SomeIpTp_91002] [

Name:	`SomeIpTp_ConfigType`
Type:	`Structure`
Range:	`implementation specific` --
Description:	This type shall contain at least all parameters that are post-build able according to chapter 10.
Available via:	`SomeIpTp.h`

] ()

8.3 Function definitions

8.3.1 SomeIpTp_GetVersionInfo

[SWS_SomeIpTp_00044] ⌈

Service name:	SomeIpTp_GetVersionInfo	
Syntax:	`void SomeIpTp_GetVersionInfo( Std_VersionInfoType* VersionInfo )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	
Available via:	`SomeIpTp.h`	

⌋ (SRS_BSW_00407,SRS_BSW_00411)

[SWS_SomeIpTp_00066] ⌈
If the parameter SomeIpTp_VersionInfoPtr of the API `SomeIpTp_GetVersionInfo()` equals NULL_PTR and if development error detection is enabled (i.e. `SomeIpTpDevErrorDetect` is set to TRUE), the function `SomeIpTp_GetVersionInfo`, the API `Det_ReportError()` shall be called with the development error code SOMEIPTP_E_PARAM_POINTER.
⌋ ()

8.3.2 SomeIpTp_Init

[SWS_SomeIpTp_00046] [

Service name:	SomeIpTp_Init	

Syntax:	`void SomeIpTp_Init(`
	`  const SomeIpTp_ConfigType* config`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	config	Base pointer to the configuration structure of the SOME/IP TP module.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the SOME/IP TP module.	
Available via:	`SomeIpTp.h`	

] (SRS_BSW_00407,SRS_BSW_00411)

Note:
The AUTOSAR ECU StateManager calls this SOME/IP TP API service with the address of the static configuration structure of the module in parameter SomeIpTp_ConfigPtr.

8.3.3 SomeIpTp_Transmit

[SWS_SomeIpTp_00047] [

Service name:	SomeIpTp_Transmit	

Syntax:	`Std_ReturnType SomeIpTp_Transmit(`
	`  PduIdType TxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x49	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Transmit request has been accepted. E_NOT_OK: Transmit request has not been accepted.
Description:	Requests transmission of a PDU.	
Available via:	`SomeIpTp.h`	

] ()

[SWS_SomeIpTp_00052] ⌈
If `SomeIpTp_Transmit()` is called before the SOME/IP TP module has been initialized with a call of `SomeIpTp_Init()`, the AP shall return with E_NOT_OK and stop the new session.
⌋()

[SWS_SomeIpTp_00048]⌈
If development error detection is enabled: `SomeIpTp_Transmit()` shall check that the service `SomeIpTp_Init()` was previously called. If the check fails, `SomeIpTp_Transmit()` shall raise the development error SOMEIPTP_E_UNINIT.
⌋ ( )

[SWS_SomeIpTp_00049] ⌈
If parameter `TxPduId` of `SomeIpTp_Transmit()` has an invalid value and if development error detection is enabled (i.e. `SomeIpTpDevErrorDetect` is set to TRUE), the API `Det_ReportError()` shall be called with the development error code SOMEIPTP_E_PARAM.
⌋()

[SWS_SomeIpTp_00050]⌈
If parameter PduInfoPtr of `SomeIpTp_Transmit()` equals NULL_PTR and if development error detection is enabled (i.e. `SomeIpTpDevErrorDetect` is set to TRUE), the API `Det_ReportError()` shall be called with the development error code SOMEIPTP_E_PARAM_POINTER.
⌋ ()

8.4 Call-back notifications

8.4.1 SomeIpTp_TriggerTransmit

[SWS_SomeIpTp_00053] ⌈

Service name:	SomeIpTp_TriggerTransmit	
Syntax:	`Std_ReturnType SomeIpTp_TriggerTransmit( PduIdType TxPduId, PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x41	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout):	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	Within this API, the upper layer module (called module) shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Available via:	`SomeIpTp.h`	

⌋ ()

[SWS_SomeIpTp_00072]⌈
If development error detection is enabled: `SomeIpTp_TriggerTransmit()` shall check that the service `SomeIpTp_Init()` was previously called. If the check fails, `SomeIpTp_TriggerTransmit()` shall raise the development error SOMEIPTP_E_UNINIT.
⌋ ( )

[SWS_SomeIpTp_00055]⌈In case the given `PduInfoPtr->SduLength` is smaller than the actual PDU-length, `SomeIpTp_TriggerTransmit()` shall not copy any data and return E_NOT_OK.
⌋()

8.4.2 SomeIpTp_RxIndication

[SWS_SomeIpTp_00056] ⌈

Service name:	SomeIpTp_RxIndication	

Syntax:	`void SomeIpTp_RxIndication(`
	`  PduIdType RxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x42	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received PDU from a lower layer communication interface module.	
Available via:	`SomeIpTp.h`	

⌋ ()

[SWS_SomeIpTp_00057]⌈
If development error detection is enabled: `SomeIpTp_RxIndication()` shall check that the service `SomeIpTp_Init()` was previously called. If the check fails, `SomeIpTp_RxIndication()` shall raise the development error SOMEIPTP_E_UNINIT.
⌋ ( )

8.4.3 SomeIpTp_TxConfirmation

[SWS_SomeIpTp_91001] ⌈

Service name:	SomeIpTp_TxConfirmation	

Syntax:	`void SomeIpTp_TxConfirmation(`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via:	`SomeIpTp.h`	

⌋ ()

[SWS_SomeIpTp_00067]⌈
If development error detection is enabled: `SomeIpTp_TxConfirmation()` shall check that the service `SomeIpTp_Init()` was previously called. If the check fails, `SomeIpTp_TxConfirmation()` shall raise the development error SOMEIPTP_E_UNINIT.
⌋ ( )

8.5 Scheduled functions

8.5.1 SomeIpTp_MainFunctionTx

[SWS_SomeIpTp_00058] [

Service name:	SomeIpTp_MainFunctionTx
Syntax:	`void SomeIpTp_MainFunctionTx( void )`
Service ID[hex]:	0x03
Description:	This function performs the processing of the AUTOSAR SOME/IP TP module's transmission activities.
Available via:	`SchM_SomeIpTp.h`

] (SRS_BSW_00373, SRS_BSW_00425)

[SWS_SomeIpTp_00059][A call to `SomeIpTp_MainFunctionTx()` shall simply return if the AUTOSAR SOME/IP TP module was not previously initialized with a call to `SomeIpTp_Init()`.](SRS_BSW_00425)

8.5.2 SomeIpTp_MainFunctionRx

[SWS_SomeIpTp_00069] [

Service name:	SomeIpTp_MainFunctionRx
Syntax:	`void SomeIpTp_MainFunctionRx( void )`
Service ID[hex]:	0x04
Description:	This function performs the processing of the AUTOSAR SOME/IP TP module's reception activities.
Available via:	`SchM_SomeIpTp.h`

] (SRS_BSW_00373, SRS_BSW_00425)

[SWS_SomeIpTp_00070][A call to `SomeIpTp_MainFunctionRx()` shall simply return if the AUTOSAR SOME/IP TP module was not previously initialized with a call to `SomeIpTp_Init()`.](SRS_BSW_00425)

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

[SWS_SomeIpTp_00060] ⌈

API function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
PduR_SomeIpTpCopyRxData	PduR_SomeIpTp.h	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.
PduR_SomeIpTpCopyTxData	PduR_SomeIpTp.h	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
PduR_SomeIpTpRxIndication	PduR_SomeIpTp.h	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.
PduR_SomeIpTpStartOfReception	PduR_SomeIpTp.h	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSduLength equal to 0.
PduR_SomeIpTpTransmit	PduR_SomeIpTp.h	Requests transmission of a PDU.
PduR_SomeIpTpTxConfirmation	PduR_SomeIpTp.h	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.

⌋ (SRS_BSW_00384)

8.6.2 Optional Interfaces

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

[SWS_SomeIpTp_00061] ⌈

API function	*Header File*	*Description*

Det_ReportError	Det.h	Service to report development errors.

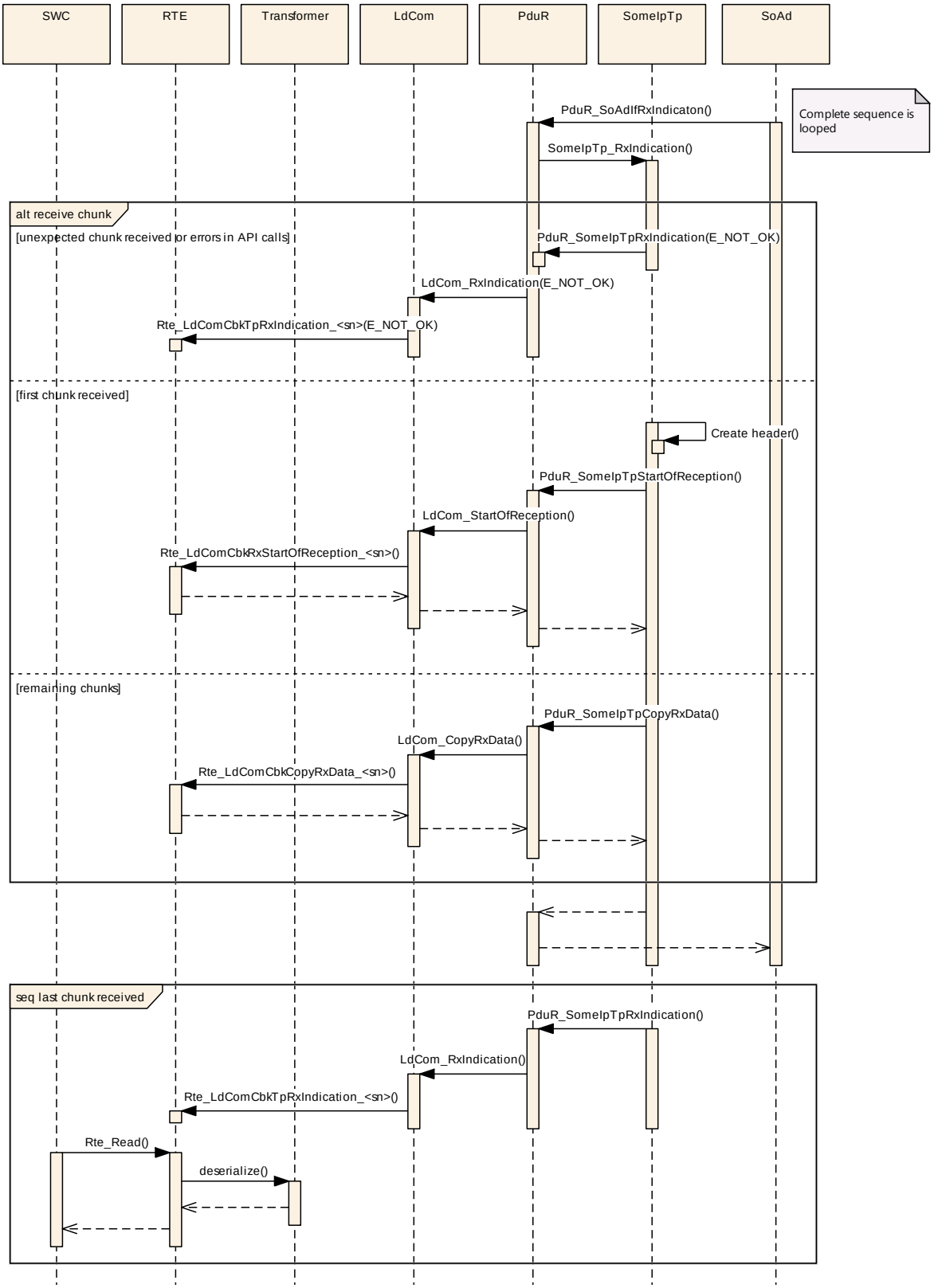
] (SRS_BSW_00384)

8.6.3 Configurable interfaces

N/A

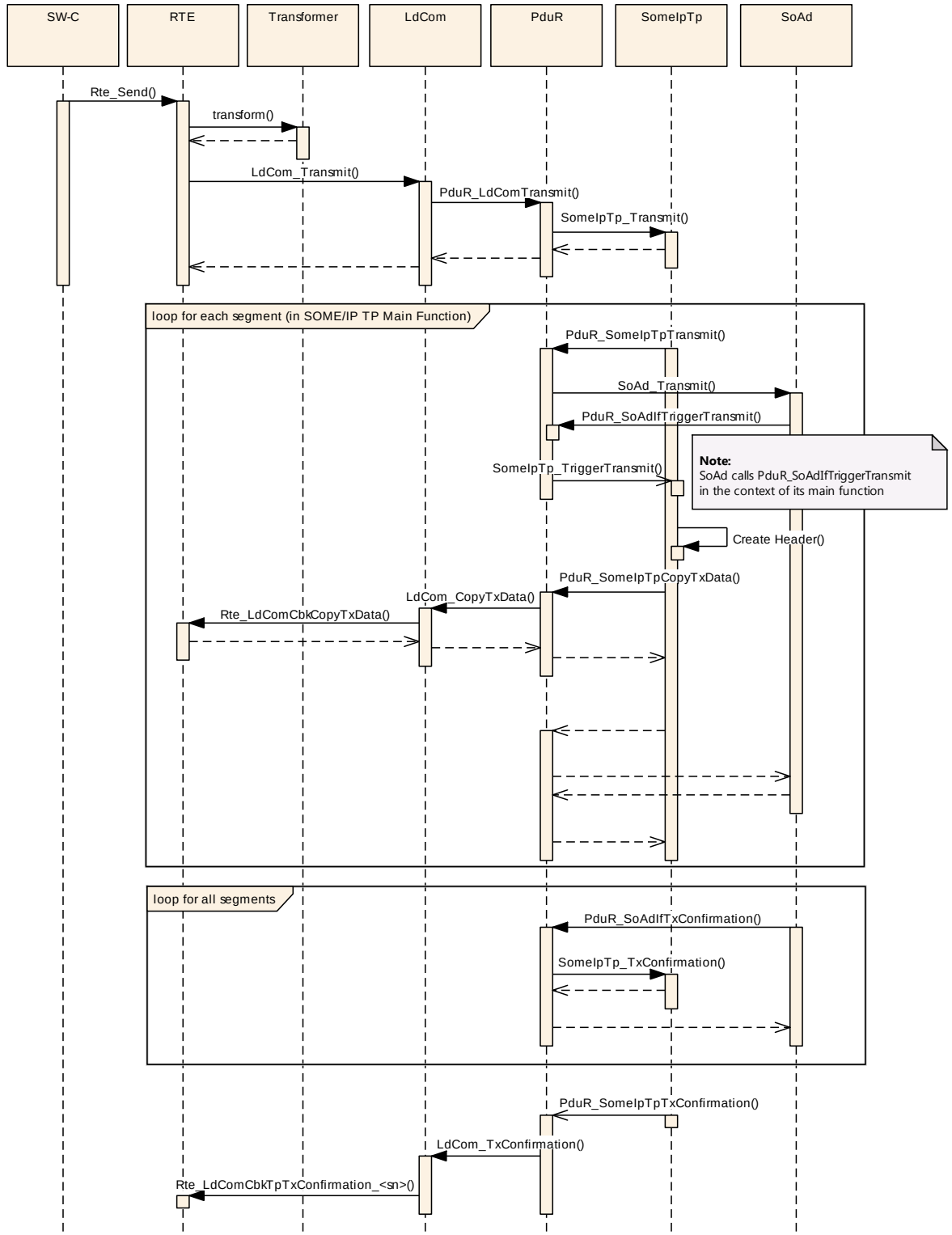
9 Sequence diagrams

9.1 Reception



Sequence 9-1 Reception of SOME/IP segments

9.2 Transmission



Sequence 9-2 Transmission of SOME/IP segments

10 Configuration specification

Chapter 10.1 specifies the structure (containers) and the parameters of the module SOME/IP TP.

Chapter 10.2 specifies additionally published information of the module SOME/IP TP.

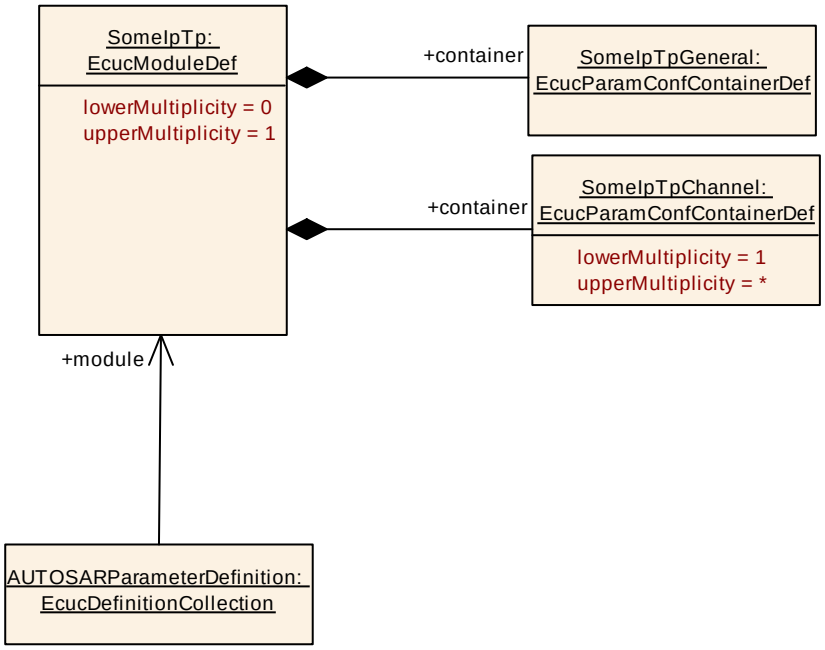
10.1 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

10.1.1 SomeIpTp

SWS Item	**ECUC_SomeIpTp_00001 :**
Module Name	*SomeIpTp*
Module Description	Configuration of the SomeIpTp module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
SomeIpTpChannel	1..*	This container contains the configuration parameters of the SomeIpTp channel.
SomeIpTpGeneral	1	This container contains the general configuration parameters of the SomeIpTp module.



10.1.2 SomeIpTpGeneral

SWS Item	**ECUC_SomeIpTp_00002 :**
Container Name	SomeIpTpGeneral
Description	This container contains the general configuration parameters of the SomeIpTp module.
Configuration Parameters	

SWS Item	**ECUC_SomeIpTp_00004 :**		
Name	SomeIpTpDevErrorDetect		
Parent Container	SomeIpTpGeneral		
Description	Switches the Development Error Detection and Notification ON or OFF.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

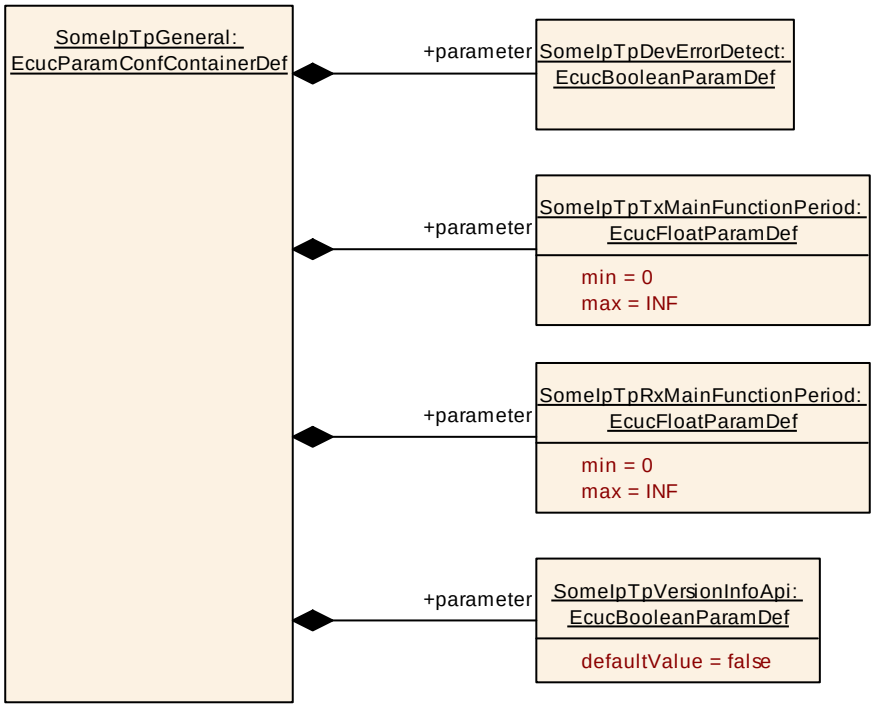
SWS Item	**ECUC_SomeIpTp_00021 :**	
Name	SomeIpTpRxMainFunctionPeriod	
Parent Container	SomeIpTpGeneral	
Description	This parameter defines the cycle time in seconds of the periodic call of the SomeIpTp_MainFunctionRx.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	--	

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

SWS Item	**ECUC_SomeIpTp_00005 :**		
Name	SomeIpTpTxMainFunctionPeriod		
Parent Container	SomeIpTpGeneral		
Description	This parameter defines the cycle time in seconds of the periodic call of the SomeIpTp_MainFunctionTx.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SomeIpTp_00019 :**		
Name	SomeIpTpVersionInfoApi		
Parent Container	SomeIpTpGeneral		
Description	Activates the SomeIpTp_GetVersionInfo() API. TRUE: Enables the SomeIpTp_GetVersionInfo() API. FALSE: SomeIpTp_GetVersionInfo() API is not included.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.1.3 SomeIpTpChannel

SWS Item	**ECUC_SomeIpTp_00003 :**		
Container Name	SomeIpTpChannel		
Description	This container contains the configuration parameters of the SomeIpTp channel.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_SomeIpTp_00006 :**		
Name	SomeIpTpNPduSeparationTime		
Parent Container	SomeIpTpChannel		
Description	Sets the duration of the minimum time in seconds the SomeIpTp module shall wait between the transmissions of N-PDUs.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_SomeIpTp_00023 :**

Name	SomeIpTpRxTimeoutTime		
Parent Container	SomeIpTpChannel		
Description	Timer to monitor the successful reception. It is started when the first NPdu is received, restarted after reception of intermediate NPdus, and is stopped when the last NPdu has been received. The value shall be calculated as follows: (SomeIpTpRxTimeoutTime = SomeIpTpNPduSeparationTime + budget), where the time budget compensates intermediary hops and jitters within the ECU implementation.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
SomeIpTpRxNSdu	0..*	The following parameters needs to be configured for each N-SDU which has to be passed as one assembled RxPdu to the upper layer.
SomeIpTpTxNSdu	0..*	The following parameters needs to be configured for each N-SDU that the SomeIpTp module transmits via the SomeIpTpChannel.

10.1.4 SomeIpTpRxNSdu

SWS Item	**ECUC_SomeIpTp_00008 :**		
Container Name	SomeIpTpRxNSdu		
Description	The following parameters needs to be configured for each N-SDU which has to be passed as one assembled RxPdu to the upper layer.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_SomeIpTp_00010 :**		
Name	SomeIpTpRxSduRef		
Parent Container	SomeIpTpRxNSdu		
Description	Reference to a Pdu in the COM-Stack that represents the assembled RxPdu which is passed via the PduR to the upper layer.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
SomeIpTpRxNPdu	1	This container contains the configuration parameters of the NPdu that is received from a lower layer

10.1.5 SomeIpTpRxNPdu

SWS Item	**ECUC_SomeIpTp_00011 :**
Container Name	SomeIpTpRxNPdu
Description	This container contains the configuration parameters of the NPdu that is received from a lower layer
Configuration Parameters	

SWS Item	**ECUC_SomeIpTp_00013 :**		
Name	SomeIpTpRxNPduHandleId		
Parent Container	SomeIpTpRxNPdu		
Description	This parameter defines the handle ID that is used by the PduR when calling SomeIpTp_RxIndication.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SomeIpTp_00012 :**		
Name	SomeIpTpRxNPduRef		
Parent Container	SomeIpTpRxNPdu		
Description	Reference to a global Pdu that is used to harmonize HandleIDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.1.6 SomeIpTpTxNSdu

SWS Item	**ECUC_SomeIpTp_00009 :**		
Container Name	SomeIpTpTxNSdu		
Description	The following parameters needs to be configured for each N-SDU that the SomeIpTp module transmits via the SomeIpTpChannel.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_SomeIpTp_00020 :**		
Name	SomeIpTpTxNSduHandleId		
Parent Container	SomeIpTpTxNSdu		
Description	This parameter defines the handle ID of the NSdu that represents the original TxSdu which is segmented and passed via the PduR to the lower layer. This handle ID is used by PduR when calling SomeIpTp_Transmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SomeIpTp_00015 :**		
Name	SomeIpTpTxNSduRef		
Parent Container	SomeIpTpTxNSdu		
Description	Reference to a global Pdu in the COM-Stack that represents the original TxSdu which is segmented and passed via the PduR to the lower layer.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
SomeIpTpTxNPdu	1	This container contains the configuration parameters of the segmented Tx NPdus that are transmitted to a lower layer.

10.1.7 SomeIpTpTxNPdu

SWS Item	**ECUC_SomeIpTp_00016 :**
Container Name	SomeIpTpTxNPdu

Description	This container contains the configuration parameters of the segmented Tx NPdus that are transmitted to a lower layer.
Configuration Parameters	

SWS Item	**ECUC_SomeIpTp_00017 :**		
Name	SomeIpTpTxNPduHandleId		
Parent Container	SomeIpTpTxNPdu		
Description	This parameter defines the handle ID that is used by PduR when calling SomeIpTp_TriggerTransmit.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_SomeIpTp_00018 :**		
Name	SomeIpTpTxNPduRef		
Parent Container	SomeIpTpTxNPdu		
Description	Reference to a global Pdu that is used to harmonize HandleIDs in the COM-Stack.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

11 Not applicable requirements

none