Document Title	Specification on SOME/IP Transport Protocol
Document Category	Software specification (SWS)
Document UID	809
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Added explicit reception handling for unsegmented SOME/IP messages and single-segment TP messages when all configured receive N-SDUs are occupied. • Clarified reporting of receive-resource limitations and the associated runtime condition. • Refined ECUC container definitions, imported datatypes, development-error classification, reception processing, and N-SDU state handling. • Removed obsolete configuration and behavioral specifications to align the document with the updated SOME/IP-TP behavior.
2025-11-27	R25-11	AUTOSAR Release Management	• Refined parallel processing of SOME/IP-TP segments • Editorial changes and bug fixes
2024-11-27	R24-11	AUTOSAR Release Management	• Several minor bugfixes • Updated Sequence diagrams for Transmission of SOME/IP segments • Editorial changes
2023-11-13	R23-11	AUTOSAR Release Management	• Several minor bugfixes • Specified behavior of PduR_SomeIpTpTransmit in case of E_NOT_OK



△

2022-11-24	R22-11	AUTOSAR Release Management	• Updated Sequence for Transmission of SOME/IP segments • Editorial changes
2021-11-25	R21-11	AUTOSAR Release Management	• Optional parameters to define a *BurstSize* to specify the number of segments that shall be transmitted in a burst and a *SeparationTime* between these bursts were added • Several minor bugfixes • Editorial changes
2020-11-30	R20-11	AUTOSAR Release Management	• Several minor bugfixes • Editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	• Editorial changes • Changed Document Status from Final to published
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 7

2 Acronyms and Abbreviations 8

3 Related documentation 9
- 3.1 Input documents & related standards and norms . . . 9
- 3.2 Related specification . . . 9

4 Constraints and assumptions 10
- 4.1 Limitations . . . 10
- 4.2 Applicability to car domains . . . 10

5 Dependencies to other modules 11
- 5.1 AUTOSAR PDU Router . . . 11
- 5.2 AUTOSAR Default Error Tracer . . . 11

6 Requirements Tracing 12

7 Functional specification 15
- 7.1 Overview of the SOME/IP header . . . 15
  - 7.1.1 Message Type Field . . . 16
  - 7.1.2 Offset Field . . . 16
  - 7.1.3 Reserved Field . . . 17
  - 7.1.4 More Segments Flag . . . 17
  - 7.1.5 Example . . . 17
- 7.2 Module Handling . . . 19
  - 7.2.1 Initialization . . . 19
- 7.3 State handling of N-SDUs . . . 20
- 7.4 Parallel processing of SOME/IP messages . . . 20
- 7.5 Segmentation of SOME/IP messages (TX Path) . . . 22
  - 7.5.1 Size of SOME/IP segments . . . 22
  - 7.5.2 Header of SOME/IP segments . . . 23
  - 7.5.3 Sending of SOME/IP segments . . . 24
  - 7.5.4 Interruption of the disassembly process . . . 26
- 7.6 Assembly of received SOME/IP messages (RX path) . . . 28
  - 7.6.1 SOME/IP segment received with Offset 0 . . . 31
  - 7.6.2 SOME/IP segment received with Offset> 0 . . . 33
  - 7.6.3 Interruption of the assembly process . . . 34
- 7.7 Error Classification . . . 38
  - 7.7.1 Development Errors . . . 38
  - 7.7.2 Runtime Errors . . . 38
  - 7.7.3 Production Errors . . . 39

8 API specification 40
8.1 Imported types . . . 40
8.2 Type definitions . . . 40
8.3 Function definitions . . . 40
8.3.1 SomeIpTp_Init . . . 41
8.3.2 SomeIpTp_DeInit . . . 41
8.3.3 SomeIpTp_Transmit . . . 42
8.3.4 SomeIpTp_TriggerTransmit . . . 43
8.3.5 SomeIpTp_RxIndication . . . 44
8.3.6 SomeIpTp_TxConfirmation . . . 44
8.4 Scheduled functions . . . 45
8.4.1 SomeIpTp_MainFunctionTx . . . 45
8.4.2 SomeIpTp_MainFunctionRx . . . 45
8.5 Expected interfaces . . . 46
8.5.1 Mandatory Interfaces . . . 46
8.5.2 Optional Interfaces . . . 47
8.5.3 Configurable interfaces . . . 47
9 Sequence diagrams 48
9.1 Reception . . . 48
9.2 Transmission . . . 49
10 Configuration specification 50
10.1 How to read this chapter . . . 50
10.2 Containers and configuration parameters . . . 50
10.2.1 SomeIpTp . . . 50
10.2.2 SomeIpTpGeneral . . . 51
10.2.3 SomeIpTpChannel . . . 53
10.2.3.1 SomeIpTpTxChannel . . . 53
10.2.3.2 SomeIpTpRxChannel . . . 55
10.2.4 SomeIpTpRxNSdu . . . 57
10.2.5 SomeIpTpRxNPdu . . . 57
10.2.6 SomeIpTpTxNSdu . . . 58
10.2.7 SomeIpTpTxNPdu . . . 60
10.3 Published Information . . . 61
A Not applicable requirements 62
B Change History 63
B.1 Change History R26-11 . . . 63
B.1.1 Added Specification Items in R26-11 . . . 63
B.1.2 Changed Specification Items in R26-11 . . . 63
B.1.3 Deleted Specification Items in R26-11 . . . 63
B.2 Change History R25-11 . . . 63
B.2.1 Added Specification Items in R25-11 . . . 63

B.2.2 Changed Specification Items in R25-11 . . . 63
B.2.3 Deleted Specification Items in R25-11 . . . 64
B.3 Change History R24-11 . . . 64
B.3.1 Added Specification Items in R24-11 . . . 64
B.3.2 Changed Specification Items in R24-11 . . . 64
B.3.3 Deleted Specification Items in R24-11 . . . 64
B.4 Change History R23-11 . . . 64
B.4.1 Added Specification Items in R23-11 . . . 64
B.4.2 Changed Specification Items in R23-11 . . . 65
B.4.3 Deleted Specification Items in R23-11 . . . 65
B.5 Change History R22-11 . . . 65
B.5.1 Added Specification Items in R22-11 . . . 65
B.5.2 Changed Specification Items in R22-11 . . . 65
B.5.3 Deleted Specification Items in R22-11 . . . 65
B.5.4 Added Constraints in R22-11 . . . 65
B.5.5 Changed Constraints in R22-11 . . . 65
B.5.6 Deleted Constraints in R22-11 . . . 65
B.5.7 Added Advisories in R22-11 . . . 66
B.5.8 Changed Advisories in R22-11 . . . 66
B.5.9 Deleted Advisories in R22-11 . . . 66

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

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

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

Table 2.1: Acronyms and Abbreviations

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[3] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[4] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[5] SOME/IP Protocol Specification
AUTOSAR_FO_PRS_SOMEIPProtocol

[6] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter

[7] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for SOME/IP Transport Protocol.

Thus, the specification SWS BSW General shall be considered as additional and required specification for SOME/IP Transport Protocol.

[1, AUTOSAR glossary] [2, SWS BSW General] [3, SRS General] [4, EXP Layered Software Architecture] [5, PRS SOME/IP Protocol] [6, SWS PDU Router]

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 Tracing

The following tables reference the requirements specified in [3] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_SomeIpTp_00083]
[SRS_BSW_00168]	SW components shall be tested by a function defined in a common API in the Basis-SW	[SWS_SomeIpTp_00083]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_SomeIpTp_00083]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_SomeIpTp_00043]
[SRS_BSW_00310]	API naming convention	[SWS_SomeIpTp_00047]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_SomeIpTp_00091]
[SRS_BSW_00337]	Classification of development errors	[SWS_SomeIpTp_00074] [SWS_SomeIpTp_00075]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_SomeIpTp_00093]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_SomeIpTp_00055]
[SRS_BSW_00360]	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	[SWS_SomeIpTp_00053] [SWS_SomeIpTp_00056] [SWS_SomeIpTp_91001]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_SomeIpTp_00074]
[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_00375]	Basic Software Modules shall report wake-up reasons	[SWS_SomeIpTp_00083]
[SRS_BSW_00383]	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	[SWS_SomeIpTp_00083]
[SRS_BSW_00384]	The Basic Software Module specifications shall specify at least in the description which other modules they require	[SWS_SomeIpTp_00060] [SWS_SomeIpTp_00061]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_SomeIpTp_00093] [SWS_SomeIpTp_00098] [SWS_SomeIpTp_00099]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00388]	Containers shall be used to group configuration parameters that are defined for the same object	[SWS_SomeIpTp_00083]
[SRS_BSW_00389]	Containers shall have names	[SWS_SomeIpTp_00083]
[SRS_BSW_00390]	Parameter content shall be unique within the module	[SWS_SomeIpTp_00083]
[SRS_BSW_00392]	Parameters shall have a type	[SWS_SomeIpTp_00083]
[SRS_BSW_00393]	Parameters shall have a range	[SWS_SomeIpTp_00083]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_SomeIpTp_00083]
[SRS_BSW_00403]	The Basic Software Module specifications shall specify for each parameter/container whether it supports different values or multiplicity in different configuration sets	[SWS_SomeIpTp_00083]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_SomeIpTp_91002]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_SomeIpTp_00076] [SWS_SomeIpTp_00090]
[SRS_BSW_00411]	AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of API(s)	[SWS_SomeIpTp_00046]
[SRS_BSW_00416]	The sequence of modules to be initialized shall be configurable	[SWS_SomeIpTp_00083]
[SRS_BSW_00417]	Software which is not part of the SW-C shall report error events only after the Dem is fully operational.	[SWS_SomeIpTp_00083]
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as `const` it should be placed into a separate c-file	[SWS_SomeIpTp_00083]
[SRS_BSW_00422]	Pre-de-bouncing of error status information is done within the Dem	[SWS_SomeIpTp_00083]
[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]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_SomeIpTp_00083]
[SRS_BSW_00452]	Classification of runtime errors	[SWS_SomeIpTp_00065] [SWS_SomeIpTp_00098] [SWS_SomeIpTp_00099]
[SRS_BSW_00461]	Modules called by generic modules shall satisfy all interfaces requested by the generic module	[SWS_SomeIpTp_00083]
[SRS_BSW_00469]	Fault detection and healing of production errors	[SWS_SomeIpTp_00083]
[SRS_BSW_00471]	Do not cause dead-locks on detection of production errors - the ability to heal from previously detected production errors	[SWS_SomeIpTp_00083]

▽

△

Requirement	Description	Satisfied by
[SRS_BSW_00472]	Avoid detection of two production errors with the same root cause.	[SWS_SomeIpTp_00083]
[SRS_BSW_00478]	Timing limits of main functions	[SWS_SomeIpTp_00083]
[SRS_BSW_00480]	Null pointer errors shall follow a naming rule	[SWS_SomeIpTp_00075]
[SRS_BSW_00481]	Invalid configuration set selection errors shall follow a naming rule	[SWS_SomeIpTp_00052]
[SRS_BSW_00490]	List possible security events	[SWS_SomeIpTp_00083]
[SRS_BSW_00491]	Specification of trigger conditions and context data	[SWS_SomeIpTp_00083]

Table 6.1: Requirements Tracing

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.

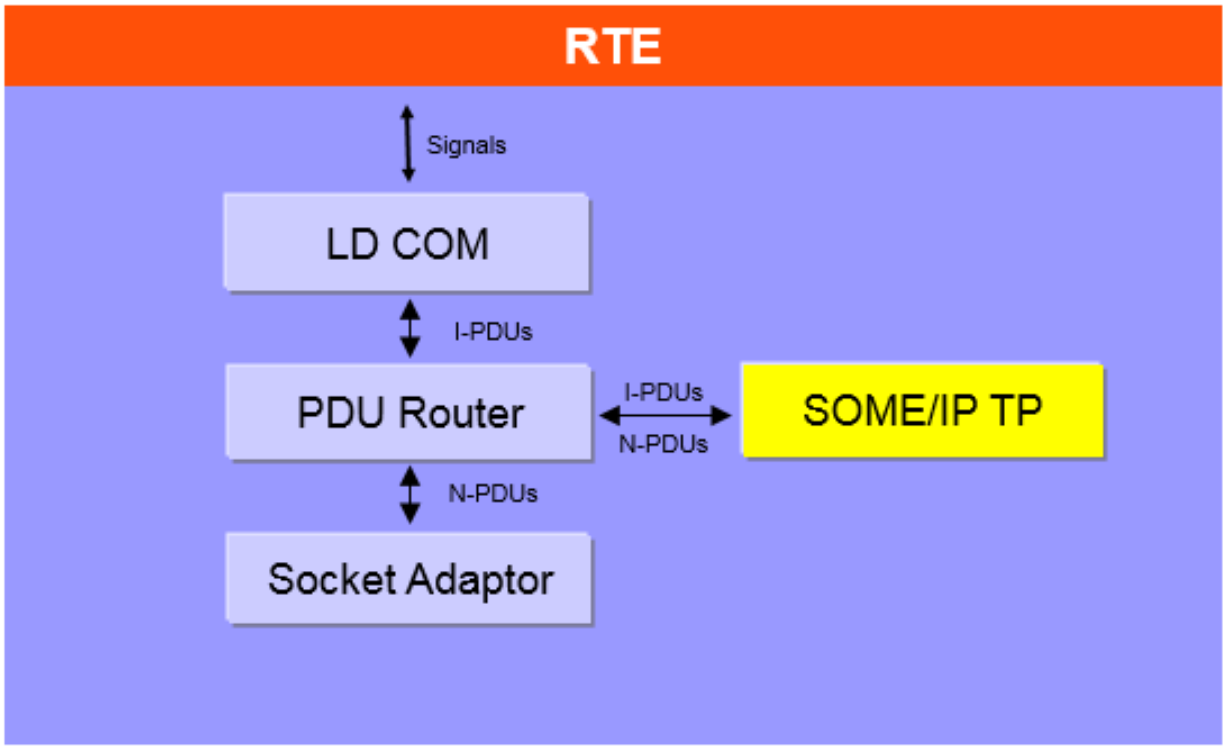

Figure 7.1: Location of the SOME/IP TP module

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.

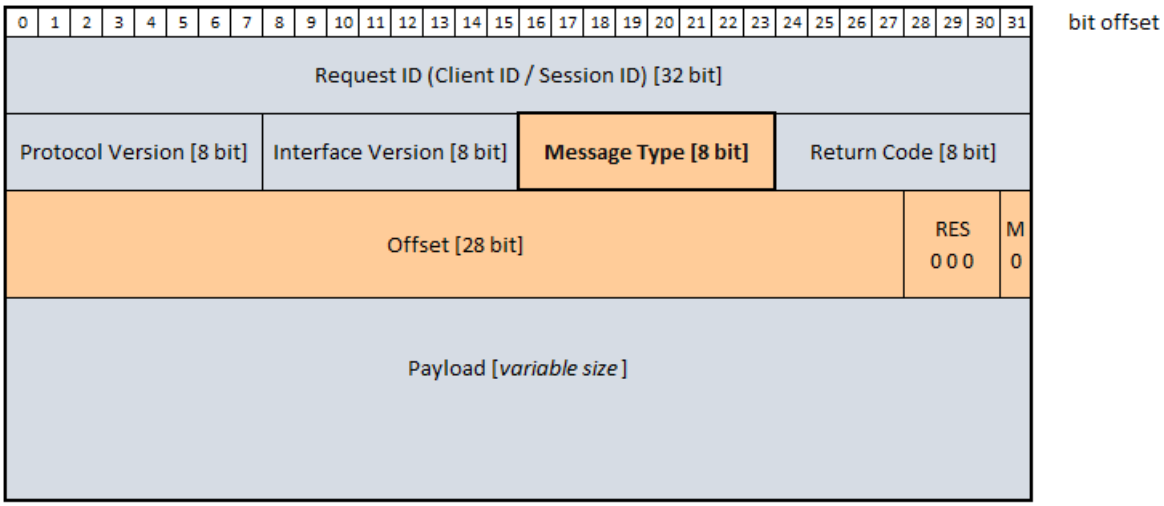

Figure 7.2: SOME/IP TP header

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

Table 7.1: 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 5880 bytes payload has to be transmitted.

The Length field of this original SOME/IP message is set to 8 + 5880 bytes.

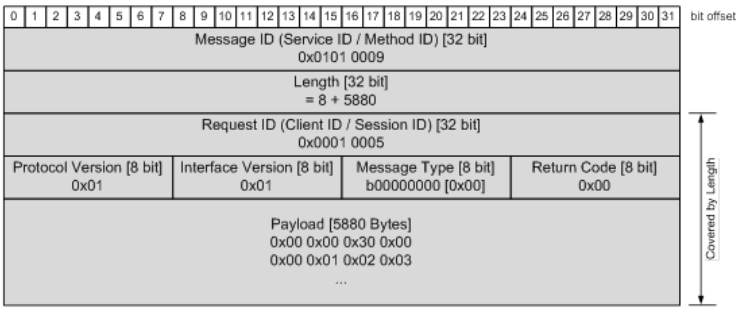

Figure 7.3: Example: Header of Original SOME/IP message

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 table 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

Table 7.2: 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’:

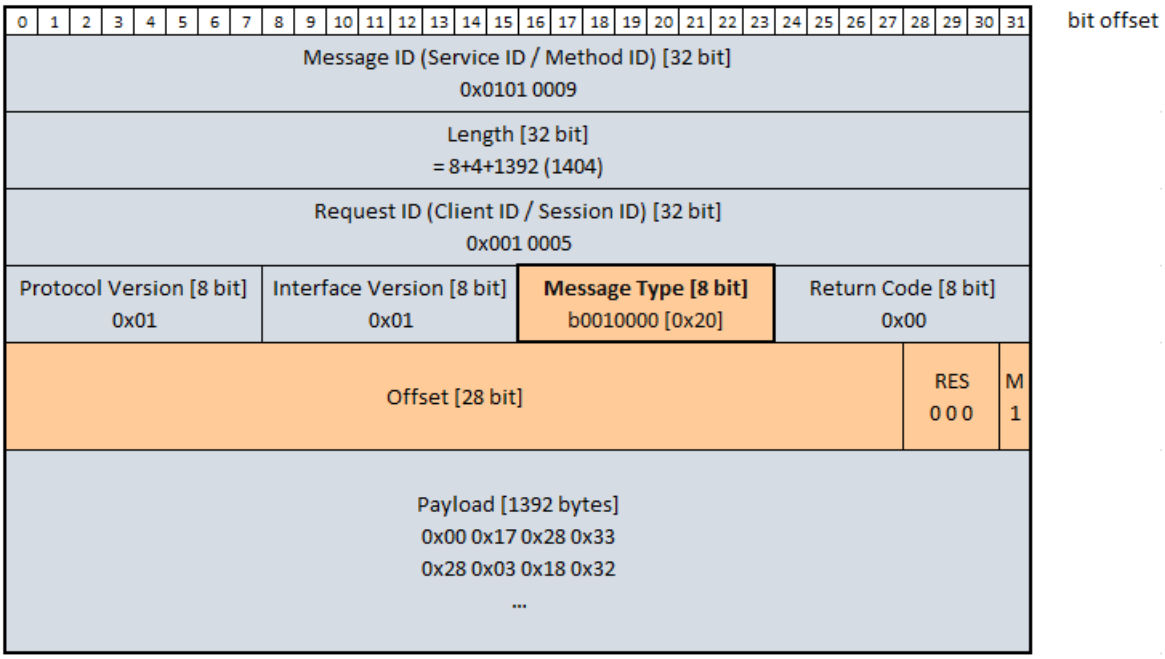

Figure 7.4: Example: Header of the SOME/IP segments

- 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’.

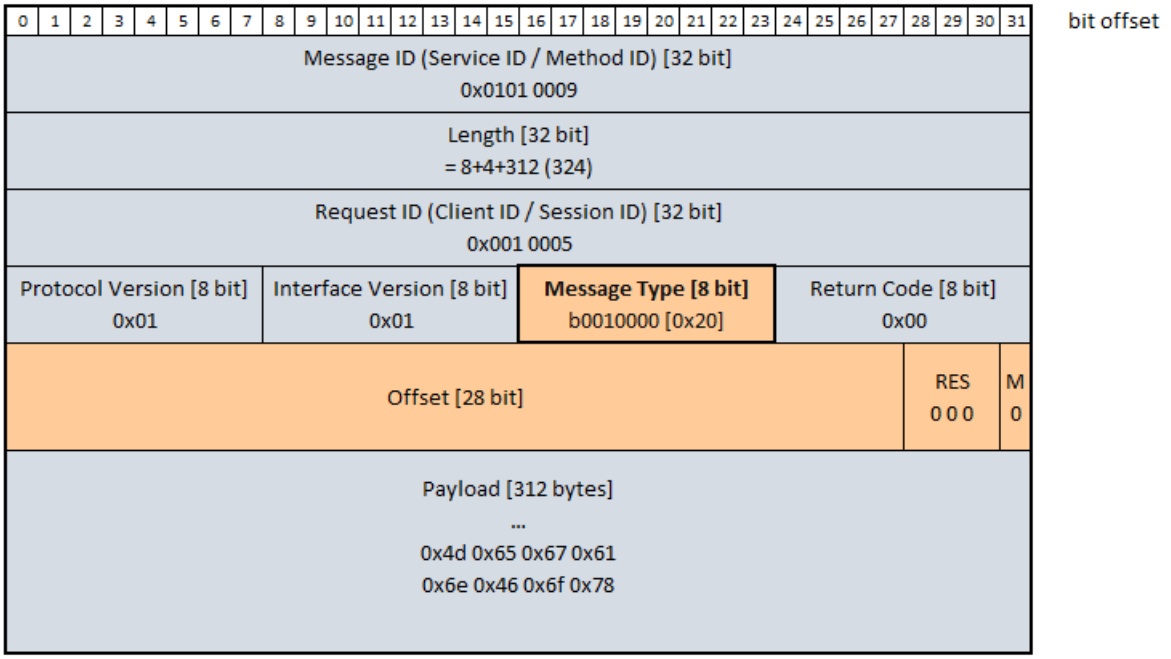

Figure 7.5: Example: Header of the last SOME/IP segment

7.2 Module Handling

This section contains description of auxiliary functionality of the SomeIpTp module.

7.2.1 Initialization

The SomeIpTp module is initialized via `SomeIpTp_Init`, and de-initialized via `SomeIpTp_DeInit`. Except for `SomeIpTp_Init`, the API functions of the SomeIpTp module may only be called after the module has been properly initialized.

[SWS_SomeIpTp_00090] Call of `SomeIpTp_Init`

Upstream requirements: SRS_BSW_00406

⌈A call to `SomeIpTp_Init` shall perform the following actions:

- Initializes all internal variables.
- Flush all internal mappings and memory sections.
- Set all configured `SomeIpTpRxNSdu` of each `SomeIpTpRxChannel` and `SomeIpTpTxNSdu` of each `SomeIpTpTxChannel` to state `NSDU_AVAILABLE`.
- Set the SomeIpTp module to initialized state.

⌋

[SWS_SomeIpTp_00091] Call of `SomeIpTp_DeInit`

Upstream requirements: SRS_BSW_00336

⌈A call to `SomeIpTp_DeInit` sets the SomeIpTp module back to the uninitialized state.⌋

[SWS_SomeIpTp_00093] Handling if `SomeIpTp_Init` is called in initialized state

Upstream requirements: SRS_BSW_00350, SRS_BSW_00386

⌈When `SomeIpTp_Init` is called in initialized state, the SomeIpTp module shall not re-initialize its internal variables, flush internal mappings and memory sections or change the state of the configured N-PDUs. It shall instead call `Det_ReportError` with the error code `SOMEIPTP_E_REINIT` if development error reporting is enabled (see `SomeIpTpDevErrorDetect`).⌋

7.3 State handling of N-SDUs

The SomeIpTp module has to maintain the usage-state of each `SomeIpTpRxNSdu` and `SomeIpTpTxNSdu` which could be configured per `SomeIpTpRxChannel` or `SomeIpTpTxChannel`. Therefore each N-SDU has two states `NSDU_IN_USE` or `NSDU_AVAILABLE`.

Note: The definition of `NSDU_IN_USE` or `NSDU_AVAILABLE` represents only the functional behavior, but not the implementation, since the state of a N-SDU is kept locally and is not propagated to other modules. Therefore, no type definition for the N-SDU state is specified.

[SWS_SomeIpTp_00094] Each N-SDU shall have a N-SDU state ⌈The SomeIpTp module shall maintain an internal usage state for each N-SDU of all configured `SomeIpTpRxNSdu` and `SomeIpTpTxNSdu` at each `SomeIpTpRxChannel` and `SomeIpTpTxChannel`, indicating whether the N-SDU is available (`NSDU_AVAILABLE`) or is use (`NSDU_IN_USE`).⌋

Note:
Please note the terms `NSDU_AVAILABLE` and `NSDU_IN_USE` only describe the functional behavior and do not mandate a specific implementation or state representation.

7.4 Parallel processing of SOME/IP messages

The SomeIpTp module configuration represents the reception and transmission of SOME/IP messages as `SomeIpTpRxChannel` and `SomeIpTpTxChannel`. Each SOME/IP TP channel has exactly one N-PDU which is used for the interaction with the lower layer modules. Each SOME/IP channel need to have a least one N-SDU which is used for the interaction with the upper layer, but could also have multiple N-

SDUs to support parallel processing of SOME/IP messages. The following parallel processing approaches are supported:

- parallel processing of SOME/IP messages that belong to different SomeIpTpRxChannels or SomeIpTpTxChannels
- parallel processing of SOME/IP messages that belong to the same SomeIpTpRxChannel or SomeIpTpTxChannel but have multiple upper layer N-SDUs configured (see SomeIpTpRxNSdu or SomeIpTpTxNSdu

A SOME/IP message is identified with the Message ID that is encoded in the first four bytes of the SOME/IP header. On reception of a SOME/IP message at the SoAd, the Message ID is extracted by the SoAd as Header ID (see SoAdRxPduHeaderId) to identify the corresponding SoAdSocketRouteDest. This SoAdSocketRouteDest has a PDU ID configured (see SoAdRxPduId) that is used to forward the remaining part of the SOME/IP message and , if configured, the SOCKET_CONNECTION_ID_16 as meta data to the SomeIpTp module. The SomeIpTp module uses the given PDU ID to identify the SomeIpTpRxNPdu where the given PDU ID and the configured SomeIpTpRxNPduHandleId match. The matching SomeIpTpRxNPdu is configured in a SomeIpTpRxChannel. This SomeIpTpRxChannel configures one or more SomeIpTpRxNSdu to address the destination module (e.g. Com) where the SOME/IP message or its segments are forwarded by the SomeIpTp module:

- For parallel processing at reception side of SOME/IP messages that belong to different SomeIpTpRxChannels, the SomeIpTp module could distinguish the parallel processing by considering different PDU IDs (SomeIpTpRxNPduHandleId). One SomeIpTpRxNPdu belongs to one SomeIpTpRxNSdu (1:1 mapping).
- For parallel processing at reception side of SOME/IP messages that belong to the same SomeIpTpRxChannel, the SOME/IP module need to select arbitrarily one of the configured and available upper layer N-SDUs (see SomeIpTpRxNSdu). Additionally, it needs to consider the transmission source of the SOME/IP message, by creating a mapping of the allocated N-SDU with the Client ID given in SOME/IP header and, if configured, by considering the SOCKET_CONNECTION_ID_16 provided via meta data. The SomeIpTp module need to maintain the usage-state of the upper layer N-SDUs (see Chapter 7.3). One SomeIpTpRxNPdu belongs to multiple SomeIpTpRxNSdus (1:n mapping).

On transmission for a SOME/IP message requested by the upper layer of the SomeIpTp module, the upper layer module provide the configured PDU ID. The SomeIpTp module uses the given PDU ID to identify the SomeIpTpTxNSdu where the given PDU ID and the configured SomeIpTpTxNSduHandleId match. The SomeIpTpTxNSdu belongs to a SomeIpTpTxChannel where exactly one SomeIpTpTxNPdu is configured that is used to forward SOME/IP message segments to the lower layer:

- For parallel processing of SOME/IP messages that belong to different SomeIpTpTxChannels, the SomeIpTp module need to maintain the usage-state of the

corresponding the upper layer N-SDU (see Chapter 7.3). One SomeIpTpTxNPdu belongs to one SomeIpTpTxNSdu (1:1 mapping).

- For parallel processing of SOME/IP messages that belong to the same SomeIpTpTxChannel, the SOME/IP module need to maintain the usage-state of the corresponding upper layer N-SDUs (see Chapter 7.3). Additionally, it needs to forward SOCKET_CONNECTION_ID_16 via meta data to the lower layer, if SOCKET_CONNECTION_ID_16 is configured. One SomeIpTpTxNPdu belongs to multiple SomeIpTpTxNSdus (1:n mapping)

On transmission side the upper layer and a proper configuration of the transmission path is responsible to support parallel processing. The upper layer is in charge to choose a N-SDU that belongs to the correct SomeIpTpTxChannel.

A system design needs to respect modelling constraints of the [7, System Template] (see [constr_9395],[TPS_SYST_02442], [constr_9396], [constr_9397])

The SomeIpTpRxChannel configure transport protocol behaviour (e.g. timing) on reception. The configuration is considered for the according SomeIpTpRxNPdu. The SomeIpTpTxChannel configure transport protocol behaviour (e.g. timing) on transmission. The configuration is considered for the according SomeIpTpTxNPdu.

The parallel processing capabilities described in this section are realized by the requirements defined in other sections of this specification. In particular, the behavioral aspects related to parallel reception, assembly, segmentation, and resource handling are specified in the corresponding SWS items in Sections 7.5 and 7.6

7.5 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.

7.5.1 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.⌋

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.⌋

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.⌋

[SWS_SomeIpTp_00004] ⌈The SOME/IP TP module shall derive the maximum possible size of the segmented SOME/IP PDUs using the parameter SomeIpTpTxNPdu Ref.⌋

[SWS_SomeIpTp_00005] ⌈The SOME/IP TP module shall generate segmented SOME/IP PDUs not larger than the size derived from the parameter SomeIpTpTxNPdu Ref.⌋

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

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_SomeIpTp Transmit() is called for each segment.⌋

7.5.2 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 based on the received PDU of the upper layer:

- Request ID [32 bit] -direct copy, see SWS_SomeIpTp_00007
- Protocol Version [8 bit] - direct copy, see SWS_SomeIpTp_00007
- Interface Version [8 bit] - direct copy, see SWS_SomeIpTp_00007
- Message Type [8 bit] - calculated value, see SWS_SomeIpTp_00008
- Return Code [8 bit] - direct copy, 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().⌋

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.⌋

[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.⌋

[SWS_SomeIpTp_00011] ⌈The Offset Field of the first SOME/IP segment shall be set to '0'.⌋

[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.⌋

[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.⌋

[SWS_SomeIpTp_00014] ⌈The SOME/IP TP module shall set the More Segment Flag to '1' except for the last SOME/IP segment.⌋

[SWS_SomeIpTp_00015] ⌈The SOME/IP TP module shall set the More Segment Flag to '0' for the last SOME/IP segment.⌋

7.5.3 Sending of SOME/IP segments

[SWS_SomeIpTp_00095] Forward meta data if configured for transmission ⌈If the API `SomeIpTp_Transmit` is called and the indicated PDU given via `TxPduId` refer to a `SomeIpTpTxNPdu` that has meta data configured, then the SOME/IP TP module shall forward the meta data.⌋

[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.⌋

[SWS_SomeIpTp_00017] Transmit request without active segmentation ⌈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).
- Set the corresponding `SomeIpTpTxNSdu` to state `NSDU_IN_USE`.
- Call the API PduR_SomeIpTpTransmit() from SomeIpTp_MainFunctionTx() using the derived PDU ID and the calculated PDU size and set the SduDataPtr to NULL_PTR.

⌋

Note:

No subsequent call to PduR_SomeIpTpTxConfirmation() shall take place since the transmission request is rejected before segmentation process started.

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

[SWS_SomeIpTp_00019] ⌈The size for consecutive SOME/IP TP segments all but not the last, shall be derived by the maximum possible size of the segmented SOME/IP PDUs using the parameter SomeIpTpTxNPduRef.⌋

[SWS_SomeIpTp_00078] Verification of available transmission data ⌈The SOME/IP TP module shall verify that the available buffer returned by PduR_SomeIpTpCopyTxData() via availableDataPtr is larger (for all but the last segment) or equal (for the last segment) size of SOME/IP TP segments.⌋

[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(). If SomeIpTpTxBurstSize is config-

ured to a value > 1 the SOME/IP TP module shall debounce for the same PDU ID only every SomeIpTpTxBurstSize segments.

⌋

[SWS_SomeIpTp_00021] Successful completion of segmented transmission ⌈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().
- Set the corresponding SomeIpTpTxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpTxConfirmation() returns

⌋

Note:

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

7.5.4 Interruption of the disassembly process

[SWS_SomeIpTp_00022] Transmit request during ongoing segmentation ⌈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.
- Set the corresponding SomeIpTpTxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpTxConfirmation() returns
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

⌋

[SWS_SomeIpTp_00082] Transmit API failure during segmentation ⌈If PduR_SomeIpTpTransmit() returns something different than E_OK during the process of ongoing segmentation.

- 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.
- Set the corresponding SomeIpTpTxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpTxConfirmation() returns
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

⌋

[SWS_SomeIpTp_00023] Transmission failure indication ⌈If the API SomeIpTp_Tx Confirmation() 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.
- Set the corresponding SomeIpTpTxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpTxConfirmation() returns
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

⌋

[SWS_SomeIpTp_00024] Insufficient available transmission data ⌈

In case the available buffer returned by PduR_SomeIpTpCopyTxData() via available-DataPtr does not satisfied the following conditions

- larger or equal to 16 bytes,
- larger (for all but the last segment) or equal (for the last segment) size of SOME/IP TP segments,

SomeIpTp module shall:

- Cancel the disassembly process for this PDU ID .
- Call the API PduR_SomeIpTpTxConfirmation() with result set to E_NOT_OK.
- Set the corresponding SomeIpTpTxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpTxConfirmation() returns
- Call the API Det_ReportRuntimeError() with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

⌋

[SWS_SomeIpTp_00025] CopyTxData failure indication ⌈If an API PduR_SomeIp TpCopyTxData()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.
- Set the corresponding SomeIpTpTxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpTxConfirmation() returns.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_DISASSEMBLY_INTERRUPT.

⌋

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

[SWS_SomeIpTp_00031] Processing of SOME/IP messages with TP flag set to '0' ⌈If SomeIpTp_RxIndication is called, the indicated PDU has TP Flag set to '0' and at least one SomeIpTpRxNSdu of the affected SomeIpTpRxChannel is in state NSDU_AVAILABLE, then the SomeIpTp module shall perform the following actions in the given order:

- set the corresponding SomeIpTpRxNSdu to state NSDU_IN_USE
- call PduR_SomeIpTpStartOfReception(), PduR_SomeIpTpCopyRxData(), and PduR_SomeIpTpRxIndication(), directly after each other providing the received indication
- set the corresponding SomeIpTpRxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpRxIndication() returns
- return without processing any further reception handling (e.g. time out handling).

⌋

[SWS_SomeIpTp_00098] Processing of SOME/IP messages when the SomeIpTpRxChannel is in state NSDU_IN_USE

Upstream requirements: SRS_BSW_00386, SRS_BSW_00452

⌈If SomeIpTp_RxIndication is called, the indicated PDU has TP Flag set to '0' and if all configured SomeIpTpRxNSdus of the affected SomeIpTpRxChannel are in state NSDU_IN_USE, report a runtime error SOMEIPTP_E_ALL_RX_NSDUS_IN_USE and return.⌋

[SWS_SomeIpTp_00026] Derivation of header information for SOME/IP messages with TP flag set to '1' ⌈If the API SomeIpTp_RxIndication is called and the indicated PDU has TP Flag set to '1', then 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]

⌋

[SWS_SomeIpTp_00071] Single-segment TP message reception ⌈If SomeIpTp_RxIndication is called and the indicated PDU carries a SOME/IP messages where the following header fields are set with the following specific values:

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

and at least one SomeIpTpRxNSdu of the affected SomeIpTpRxChannel is in state NSDU_AVAILABLE, then SomeIpTp module shall perform the following actions in the given order:

- set the corresponding SomeIpTpRxNSdu to state NSDU_IN_USE
- call PduR_SomeIpTpStartOfReception(), PduR_SomeIpTpCopyRxData(), and PduR_SomeIpTpRxIndication(), directly after each other providing the received indication
- set the corresponding SomeIpTpRxNSdu to state NSDU_AVAILABLE, after PduR_SomeIpTpRxIndication() returns
- return without processing any further reception handling (e.g. time out handling).

⌋

[SWS_SomeIpTp_00099] Processing of SOME/IP messages with TP flag set to '1' but SomeIpTpRxChannel is in state NSDU_IN_USE

Upstream requirements: SRS_BSW_00386, SRS_BSW_00452

⌈If SomeIpTp_RxIndication is called and the indicated PDU carries a SOME/IP message where the following header fields are set with the following specific values:

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

and if all configured SomeIpTpRxNSdu of the affected SomeIpTpRxChannel are in state NSDU_IN_USE, report a runtime error SOMEIPTP_E_ALL_RX_NSDUS_IN_USE and return.⌋

[SWS_SomeIpTp_00096] Support parallel processing of the same PDU ID based on the identified transmission sources ⌈If the API SomeIpTp_RxIndication is called, the SOME/IP messages has TP flag set to '1' and at least one SomeIpTpRx-NSdu of the affected SomeIpTpRxChannel is in state NSDU_AVAILABLE, then the SomeIpTp module shall support parallel processing of SOME/IP messages for the same PDU ID (i.e. Message ID) based on the following rules to identify the transmission source:

- If the indicated PDU given via RxPduId has SOCKET_CONNECTION_ID_16 configured, then the SomeIpTp module shall consider SOCKET_CONNECTION_ID_16 and the Client ID to identify the transmission source.
- If the indicated PDU given via RxPduId has no SOCKET_CONNECTION_ID_16 configured, then the SomeIpTp module shall consider the Client ID to identify the transmission source.

⌋

Note:

- An indicated PDU given with the PDU ID corresponds to a Message ID.
- A call of the same Message ID (e.g. method call) at the same ECU from different clients are queued by the RTE on the transmitting ECU, since the PDU is blocked until the call of the first client is finalized. Thus, only one outstanding call from one client with the same transmission source need to be considered at the receiving ECU.
- A call of the same Message ID (e.g. method call) from different ECUs could overlap at the receiving ECU, therefore a parallel processing within the reception path at the SomeIpTp module is needed, which require to consider the information of the transmission source.

[SWS_SomeIpTp_00027] ⌈The SOME/IP TP module shall be able to store the value of the Offset Field for every PDU ID per identified transmission source ([SWS_SomeIpTp_00096]) separately.⌋

[SWS_SomeIpTp_00028] ⌈The SOME/IP TP module shall be able to store the number of Payload bytes for every PDU ID per identified transmission source ([SWS_SomeIpTp_00096]) separately which has been passed by a call of SomeIpTp_RxIndication().⌋

[SWS_SomeIpTp_00029] ⌈The SOME/IP TP module shall store the status of the More Segment Flag for every PDU ID per identified transmission source ([SWS_SomeIpTp_00096]) separately which is passed by a call of SomeIpTP_RxIndication().⌋

[SWS_SomeIpTp_00030] ⌈The SOME/IP TP module shall buffer the pointer to the Meta-data for every PDU ID per identified transmission source ([SWS_SomeIpTp_00096]) separately which is passed by the PduInfoPtr parameter of the API SomeIp Tp_RxIndication(), and forward this information when PduR_SomeIpTpStartOfReception is called.⌋

7.6.1 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 per identified transmission source ([SWS_SomeIpTp_00096]) 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).⌋

[SWS_SomeIpTp_00033] First segment reception handling ⌈If a SOME/IP segment is successfully received with Offset Field set to 0 via a configured `SomeIpTpRxNPdu` and at least one `SomeIpTpRxNSdu` of the affected `SomeIpTpRxChannel` is in state `NSDU_AVAILABLE`, then the SOME/IP TP module shall perform the following actions. Otherwise, if all configured `SomeIpTpRxNSdu`s of the affected `SomeIpTpRxChannel` are in state `NSDU_IN_USE`, report a runtime error `SOMEIPTP_E_ALL_RX_NSDUS_IN_USE` and return:

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

⌋

Note:

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

[SWS_SomeIpTp_00097] First segment reception during ongoing assembly ⌈If a SOME/IP segment is successfully received with Offset Field set to 0 via a configured `SomeIpTpRxNPdu` and the following conditions are fulfilled:

- the corresponding `SomeIpTpRxNSdu` of the affected `SomeIpTpRxChannel` is in use for a pending finalization of an ongoing assembly process
- the identified transmission source of the received PDU ID matches to the transmission source that refers to the ongoing assembly process

then the SOME/IP TP module shall perform the following actions:

- Cancel the ongoing assembly process by calling PduR_SomeIpTpRxIndication() for this PDU ID with result set to `E_NOT_OK`

- Re-start the Rx timeout time defined by SomeIpTpRxTimeoutTime of the affected SomeIpTpRxChannel.
- Start a new assembly process by calling PduR_SomeIpStartOfReception() for this PDU ID.

⌋

[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 size returned via bufferSizePtr.

If the returned size 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 SOME/IP header (excluding the SOME/IP TP header) of the assembled SOME/IP message to the SOME/IP TP's upper layer. This shall include 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]

⌋

[SWS_SomeIpTp_00079] ⌈

After calling PduR_SomeIpTpCopyRxData() to pass the SOME/IP header (excluding the SOME/IP TP header) of the assembled SOME/IP message to the SOME/IP TP's upper layer (see [SWS_SomeIpTp_00034]), the SOME/IP TP module shall call the API PduR_SomeIpTpCopyRxData() again, to provide the payload of the assembled SOME/IP message.

⌋

Note: Sequential calls of PduR_SomeIpTpCopyRxData() avoid storing of the SOME/IP TP segment in the SOME/IP TP module and support a proper handling to strip off the SOME/IP TP header by skipping 4 bytes that include the Offset field, Reserved Field and the more Segment flag.

[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.⌋

[SWS_SomeIpTp_00036] ⌈The SOME/IP TP module shall store the number of Payload bytes for every PDU ID per identified transmission source ([SWS_SomeIpTp_00096]) separately which has been passed to the upper layer.⌋

Note:

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

7.6.2 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) for each PDU ID per identified transmission source ([SWS_SomeIpTp_00096]):

- 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.⌋

[SWS_SomeIpTp_00038] ⌈The SOME/IP TP module shall store the number of Payload bytes for every PDU ID per identified transmission source ([SWS_SomeIpTp_00096]) separately which has been passed to the upper layer.⌋

[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 bufferSize Ptr provided by the previous call of the API PduR_SomeIpTpCopyRxData()is greater or equal to the received payload, call the API PduR_SomeIpTpCopyRxData()with Sdu Length set to the received Payload bytes.⌋

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] Handling of last SOME/IP TP segment that corresponds to an ongoing assembly process ⌈If a SOME/IP segment is successfully received

with the More Segment Flag set to '0' via a configured SomeIpTpRxNPdu and the indicated PDU ID in combination with its identified transmission source (see [SWS_SomeIpTp_00096]) matches to an ongoing assembly process where the corresponding SomeIpTpRxNSdu is in state NSDU_IN_USE, then the SOME/IP TP module shall perform the following actions:

- 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).
- Set the corresponding SomeIpTpRxNSdu to state NSDU_AVAILALBE, after PduR_SomeIpTpRxIndication() returns.

⌋

7.6.3 Interruption of the assembly process

[SWS_SomeIpTp_00041] Handling if SomeIpTpRxTimeoutTime expires ⌈If the Rx timeout time defined by SomeIpTpRxTimeoutTime of a SomeIpTpRxChannel expires,

- The current assembly process shall be interrupted as defined by SWS_SomeIp Tp_00054.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋

[SWS_SomeIpTp_00042] Handling if SOME/IP TP segment is received with Offset Value > 0, but no session for this PDU ID and its identified transmission source is running ⌈If the API SomeIpTp_RxIndication() is called with the Offset Value is > 0 but no session for the given PDU ID and its identified transmission source (see [SWS_SomeIpTp_00096]) 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.

⌋

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

[SWS_SomeIpTp_00054] Handling if an assembly process is interrupted ⌈If the SOME/IP TP module interrupts the assembly process because of a detected error, the SOME/IP TP module shall consider the following points:

- call PduR_SomeIpTpRxIndication() with result set to `E_NOT_OK` for the PDU ID that refers to the used `SomeIpTpRxNSdu` of the affected `SomeIpTpRxChannel`.
- Set the corresponding `SomeIpTpRxNSdu` to state `NSDU_AVAILABLE`, after PduR_SomeIpTpRxIndication() returns.
- The Rx timeout time defined by SomeIpTpRxTimeoutTime shall be canceled (if still running) for this assembly process that corresponds to the PDU ID and its identified transmission source (see [SWS_SomeIpTp_00096]).

⌋

Note: The possible reasons for interruptions are listed below.

[SWS_SomeIpTp_00062] Handling if inconsistency of received SOME/IP TP headers is detected ⌈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) per identified transmission source ([SWS_SomeIpTp_00096]),

- The current assembly process for the affected PDU of the corresponding identified transmission source ([SWS_SomeIpTp_00096]) 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.

⌋

[SWS_SomeIpTp_00045] Handling if received SOME/IP TP header has set TP-Flag to '0' for an currently active session ⌈If the API SomeIpTp_RxIndication() is called and a session for the affected PDU of the identified transmission source ([SWS_SomeIpTp_00096]) 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 for the affected PDU of the corresponding identified transmission source ([SWS_SomeIpTp_00096]) 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.

⌋

[SWS_SomeIpTp_00080] Header Inconsistency check before TP-Flag check ⌈

Before checking the TP-Flag of the Message, as a condition to interrupt the assembly process, (see [SWS_SomeIpTp_00045]), the SOME/IP TP module shall check for inconsistencies of the received SOME/IP TP headers according to [SWS_SomeIpTp_00062].⌋

[SWS_SomeIpTp_00063] The SomeIpTp module, shall check received SOME/IP TP segments with More Segment Flag set '1', whether the length of the received payload is divisible by 16 ⌈If the API SomeIpTp_RxIndication() is called, the SOME/IP TP module shall check the affected PDU per identified transmission source ([SWS_SomeIpTp_00096]) whether the length of received payload is divisible by 16 in case the More Segment Flag is set to '1'.

If the received payload bytes are not divisible by 16 in this case,

- The current assembly process for the affected PDU of the corresponding identified transmission source ([SWS_SomeIpTp_00096]) 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.

⌋

[SWS_SomeIpTp_00064] Handling if value of the received Offset Value in units of 16 bytes do not match to the sum of the received Payload bytes of the previous SOME/IP TP segments ⌈If the API SomeIpTp_RxIndication() is called, the SOME/IP TP module shall check the value of the Offset Field at the corresponding PDU per identified transmission source ([SWS_SomeIpTp_00096]. 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 for the affected PDU of the corresponding identified transmission source ([SWS_SomeIpTp_00096]) 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.

⌋

[SWS_SomeIpTp_00048] Handling if value of the received Offset Value equals '0' while the received Payload bytes of the previous SOME/IP segments is greater than '0' ⌈If the API SomeIpTp_RxIndication() is called, the SOME/IP TP module shall check the value of the Offset Field at corresponding PDU per identified transmission source ([SWS_SomeIpTp_00096]). 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 for the affected PDU of the corresponding identified transmission source ([SWS_SomeIpTp_00096]) 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

⌋

[SWS_SomeIpTp_00049] Handling if provided Rx buffer size is smaller than the required size ⌈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 for the affected PDU of the corresponding identified transmission source ([SWS_SomeIpTp_00096]) 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.

⌋

[SWS_SomeIpTp_00050] Handling if PduR_SomeIpTpCopyRxData() returns something different than `BUFREQ_OK` ⌈If the API PduR_SomeIpTpCopyRx Data()returns something else than BUFREQ_OK,

- The assembly process for this PDU of the corresponding identified transmission source 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.

⌋

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

- The assembly process for this PDU of the corresponding identified transmission source shall be stopped.
- The Rx timeout time defined by `SomeIpTpRxTimeoutTime` shall be canceled (if still running) for this assembly process that corresponds to the PDU ID and its identified transmission source (see [SWS_SomeIpTp_00096]).
- Set the corresponding `SomeIpTpRxNSdu` to state `NSDU_AVAILABLE`.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

⌋

7.7 Error Classification

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

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

7.7.1 Development Errors

[SWS_SomeIpTp_00052] Definition of development errors in module SomeIpTp

Upstream requirements: SRS_BSW_00481

⌈

Type of error	*Related error code*	*Error value*
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
Invalid configuration set selection	SOMEIPTP_E_INIT_FAILED	0x04
SOME/IP TP module is already initialized	SOMEIPTP_E_REINIT	0x05

⌋

7.7.2 Runtime Errors

[SWS_SomeIpTp_00065] Definition of runtime errors in module SomeIpTp

Upstream requirements: SRS_BSW_00452

⌈

Type of error	*Related error code*	*Error value*
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
All configured SomeIpTpRxNSdu of the affected SomeIpTpRxChannel are in state NSDU_IN_USE, while a further request is received.	SOMEIPTP_E_ALL_RX_NSDUS_IN_USE	0x09

⌋

Note: In reference to run-time error "SOMEIPTP_E_MESSAGE_TYPE" no DET will be reported for unsegmented message and is passed to the upper layer without further handling.

7.7.3 Production Errors

There are no production errors.

8 API specification

8.1 Imported types

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

[SWS_SomeIpTp_00043] Definition of imported datatypes of module SomeIpTp

Upstream requirements: SRS_BSW_00301

⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TpDataStateType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

[SWS_SomeIpTp_91002] Definition of datatype SomeIpTp_ConfigType

Upstream requirements: SRS_BSW_00404

⌈

Name	SomeIpTp_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
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

This chapter contains all APIs provided by the module.

8.3.1 SomeIpTp_Init

[SWS_SomeIpTp_00046] Definition of API function SomeIpTp_Init

Upstream requirements: SRS_BSW_00411

⌈

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	

⌋

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_Config Ptr.

8.3.2 SomeIpTp_DeInit

[SWS_SomeIpTp_91003] Definition of API function SomeIpTp_DeInit ⌈

Service Name	SomeIpTp_DeInit

Syntax	`void SomeIpTp_DeInit (`
	`  void`
	`)`

Service ID [hex]	0x05
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This function resets the SomeIpTp module to the uninitialized state.
Available via	SomeIpTp.h

⌋

8.3.3 SomeIpTp_Transmit

[SWS_SomeIpTp_00047] Definition of API function SomeIpTp_Transmit

Upstream requirements: SRS_BSW_00310

⌈

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_00076]

Upstream requirements: SRS_BSW_00406

⌈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_00074]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00369

⌈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_00075]

Upstream requirements: SRS_BSW_00337, SRS_BSW_00480

⌈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.⌋

8.3.4 SomeIpTp_TriggerTransmit

[SWS_SomeIpTp_00053] Definition of callback function SomeIpTp_TriggerTransmit

Upstream requirements: SRS_BSW_00360

⌈

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_00055]

Upstream requirements: SRS_BSW_00357

⌈In case the given PduInfoPtr->SduLength is smaller than the computed size of the SOME/IP-TP segment (considering header and payload), SomeIpTp_TriggerTransmit() shall not copy any data and return E_NOT_OK.⌋

8.3.5 SomeIpTp_RxIndication

[SWS_SomeIpTp_00056] Definition of callback function SomeIpTp_RxIndication

Upstream requirements: SRS_BSW_00360

⌈

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	

⌋

8.3.6 SomeIpTp_TxConfirmation

[SWS_SomeIpTp_91001] Definition of callback function SomeIpTp_TxConfirmation

Upstream requirements: SRS_BSW_00360

⌈

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 successfully. `E_NOT_OK`: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	

▽

△

Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
Available via	SomeIpTp.h

⌋

8.4 Scheduled functions

8.4.1 SomeIpTp_MainFunctionTx

[SWS_SomeIpTp_00058] Definition of scheduled function SomeIpTp_MainFunctionTx

Upstream requirements: SRS_BSW_00373, SRS_BSW_00425

⌈

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

⌋

[SWS_SomeIpTp_00059]

Upstream requirements: SRS_BSW_00425

⌈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().⌋

8.4.2 SomeIpTp_MainFunctionRx

[SWS_SomeIpTp_00069] Definition of scheduled function SomeIpTp_MainFunctionRx

Upstream requirements: SRS_BSW_00373, SRS_BSW_00425

⌈

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

⌋

[SWS_SomeIpTp_00070]

Upstream requirements: SRS_BSW_00425

⌈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().⌋

8.5 Expected interfaces

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

8.5.1 Mandatory Interfaces

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

[SWS_SomeIpTp_00060] Definition of mandatory interfaces required by module SomeIpTp

Upstream requirements: SRS_BSW_00384

⌈

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 reception was successful or not.

▽

△

API Function	Header File	Description
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 TpSdu Length 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.

⌋

8.5.2 Optional Interfaces

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

[SWS_SomeIpTp_00061] Definition of optional interfaces requested by module SomeIpTp

Upstream requirements: SRS_BSW_00384

⌈

API Function	Header File	Description
Det_ReportError	Det.h	Service to report development errors.

⌋

8.5.3 Configurable interfaces

N/A

9 Sequence diagrams

9.1 Reception

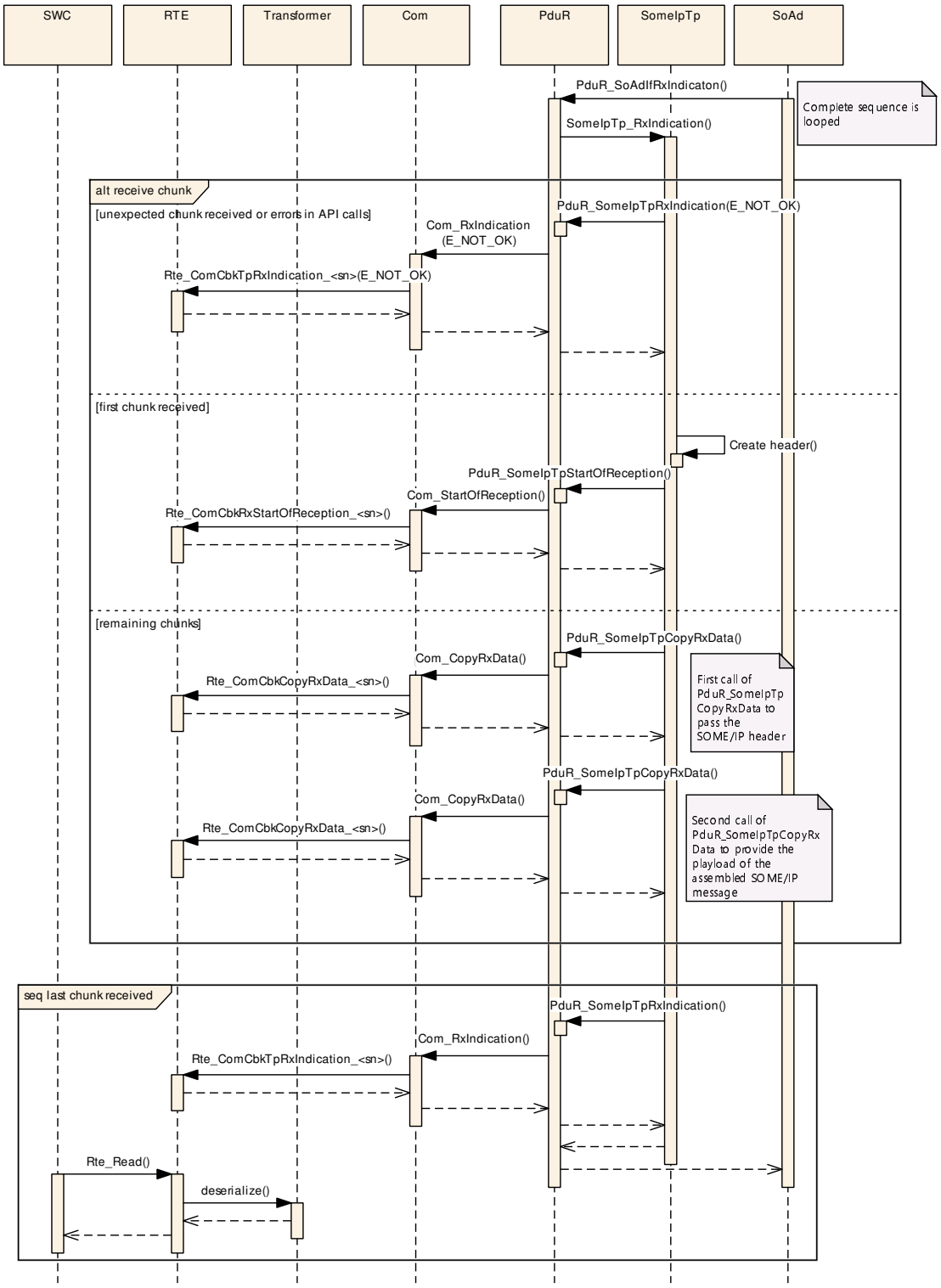

Figure 9.1: Reception of SOME/IP segments

9.2 Transmission

Sequence 9.2 depicts a sequence where the call to PduR_SomeIpTpTransmit() for the first segment according to SWS_SomeIpTp_00017 is deferred to the SomeIpTp_MainFunction().

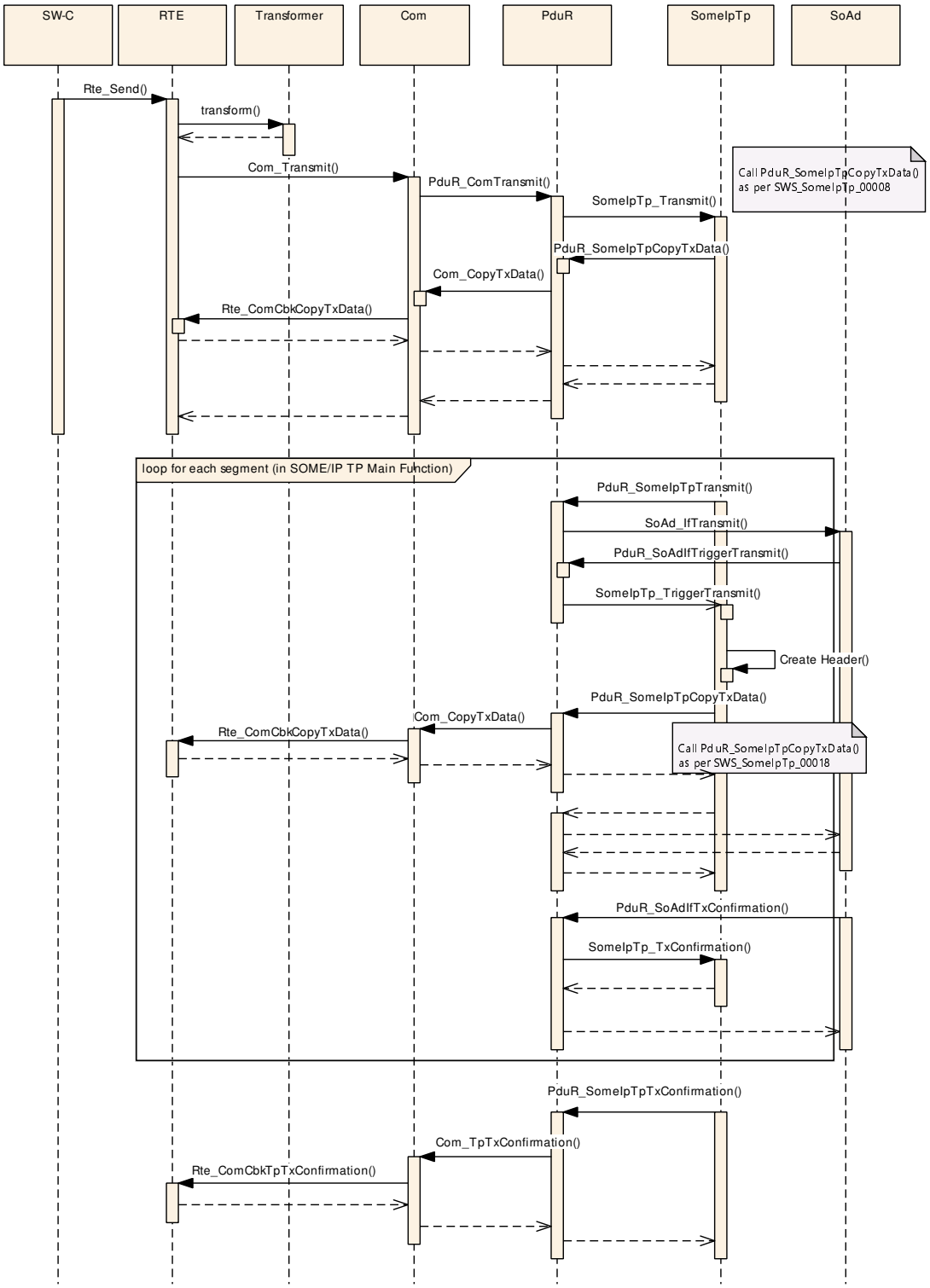

Figure 9.2: Transmission of SOME/IP segments

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module SOME/IP TP.

Chapter 10.3 specifies published information of the module SOME/IP TP.

10.1 How to read this chapter

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

10.2 Containers and configuration parameters

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

10.2.1 SomeIpTp

[ECUC_SomeIpTp_00001] Definition of EcucModuleDef SomeIpTp ⌈

Module Name	SomeIpTp
Description	Configuration of the SomeIpTp module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
SomeIpTpGeneral	1	This container contains the general configuration parameters of the SomeIpTp module.
SomeIpTpRxChannel	0..*	This container contains the configuration parameters of the SomeIpTp reception channel.
SomeIpTpTxChannel	0..*	This container contains the configuration parameters of the SomeIpTp transmission channel.

⌋

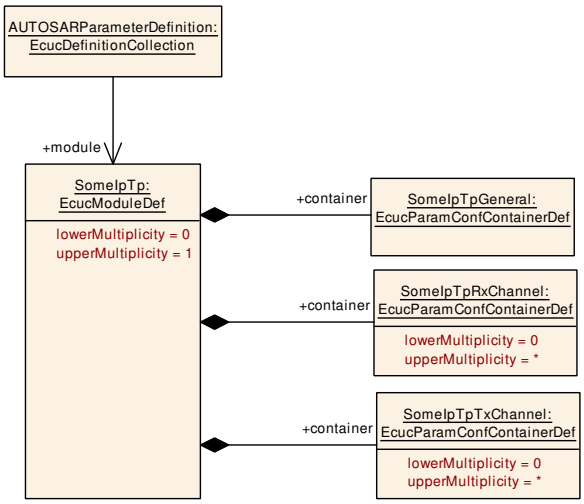

Figure 10.1

10.2.2 SomeIpTpGeneral

[ECUC_SomeIpTp_00002] Definition of EcucParamConfContainerDef SomeIpTp General ⌈

Container Name	SomeIpTpGeneral
Parent Container	SomeIpTp
Upstream mapping defined	–
Description	This container contains the general configuration parameters of the SomeIpTp module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SomeIpTpDevErrorDetect	1	[ECUC_SomeIpTp_00004]
SomeIpTpRxMainFunctionPeriod	1	[ECUC_SomeIpTp_00021]
SomeIpTpTxMainFunctionPeriod	1	[ECUC_SomeIpTp_00005]

No Included Containers

⌋

[ECUC_SomeIpTp_00004] Definition of EcucBooleanParamDef SomeIpTpDevErrorDetect ⌈

Parameter Name	SomeIpTpDevErrorDetect
Parent Container	SomeIpTpGeneral
Upstream mapping defined	–

▽

△

Description	Switches the Development Error Detection and Notification ON or OFF.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_SomeIpTp_00021] Definition of EcucFloatParamDef SomeIpTpRxMainFunctionPeriod ⌈

Parameter Name	SomeIpTpRxMainFunctionPeriod	
Parent Container	SomeIpTpGeneral	
Upstream mapping defined	–	
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	–	
Pre-Compile only	true	

⌋

[ECUC_SomeIpTp_00005] Definition of EcucFloatParamDef SomeIpTpTxMainFunctionPeriod ⌈

Parameter Name	SomeIpTpTxMainFunctionPeriod	
Parent Container	SomeIpTpGeneral	
Upstream mapping defined	–	
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	–	
Pre-Compile only	true	

⌋

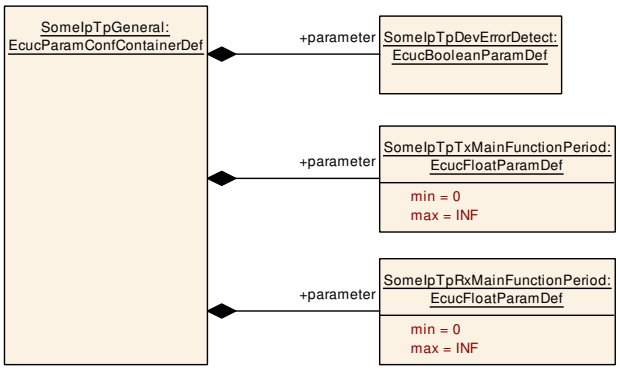

Figure 10.2

10.2.3 SomeIpTpChannel

10.2.3.1 SomeIpTpTxChannel

[ECUC_SomeIpTp_00025] Definition of EcucParamConfContainerDef SomeIpTpTxChannel ⌈

Container Name	SomeIpTpTxChannel	
Parent Container	SomeIpTp	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	This container contains the configuration parameters of the SomeIpTp transmission channel.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SomeIpTpNPduSeparationTime	1	[ECUC_SomeIpTp_00006]
SomeIpTpTxBurstSize	0..1	[ECUC_SomeIpTp_00024]

Included Containers		
Container Name	**Multiplicity**	**Description**
SomeIpTpTxNPdu	1	This container contains the configuration parameters of the segmented Tx NPdus that are transmitted to a lower layer.
SomeIpTpTxNSdu	1..*	This container defines the upper layer Sdus to be transmitted in the context of the SomeIpTpTxChannel. This PDU can forward meta data items of type SOCKET_CONNECTION_ID_16.

⌋

[ECUC_SomeIpTp_00006] Definition of EcucFloatParamDef SomeIpTpNPduSeparationTime ⌈

Parameter Name	SomeIpTpNPduSeparationTime	
Parent Container	SomeIpTpTxChannel	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

[ECUC_SomeIpTp_00024] Definition of EcucIntegerParamDef SomeIpTpTxBurstSize ⌈

Parameter Name	SomeIpTpTxBurstSize	
Parent Container	SomeIpTpTxChannel	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	Specifies the number of segments SomeIpTp shall transmit without applying the SomeIpTpNPduSeparationTime.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	1 .. 18446744073709551615	
Default value	1	
Post-Build Multiplicity	-	Selectable
Post-Build Value	Loadable	Selectable

⌋

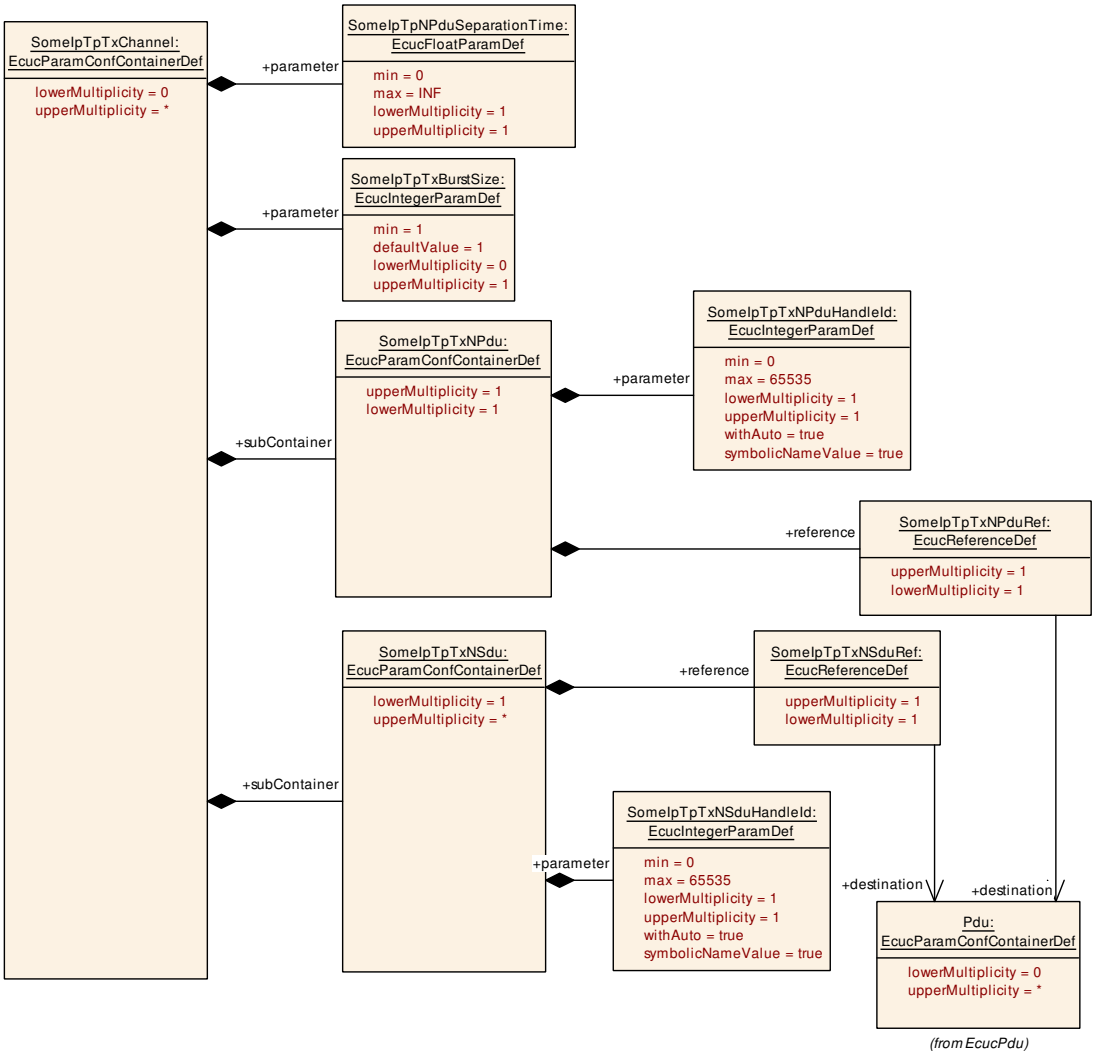

Figure 10.3

10.2.3.2 SomeIpTpRxChannel

[ECUC_SomeIpTp_00026] Definition of EcucParamConfContainerDef SomeIpTp RxChannel ⌈

Container Name	SomeIpTpRxChannel	
Parent Container	SomeIpTp	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	This container contains the configuration parameters of the SomeIpTp reception channel.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SomeIpTpRxTimeoutTime	1	[ECUC_SomeIpTp_00023]

Included Containers		
Container Name	**Multiplicity**	**Description**
SomeIpTpRxNPdu	1	This container contains the configuration parameters of the NPdu that is received from a lower layer
SomeIpTpRxNSdu	1..*	This container defines the upper layer Sdus assembled in the context of the SomeIpTpRxChannel. This PDU can forward meta data items of type SOCKET_CONNECTION_ID_16.

⌋

[ECUC_SomeIpTp_00023] Definition of EcucFloatParamDef SomeIpTpRxTimeoutTime ⌈

Parameter Name	SomeIpTpRxTimeoutTime	
Parent Container	SomeIpTpRxChannel	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	Timer to monitor the successful reception (see FO_PRS_SOMEIP_00378). 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 Value	Loadable	Selectable

⌋

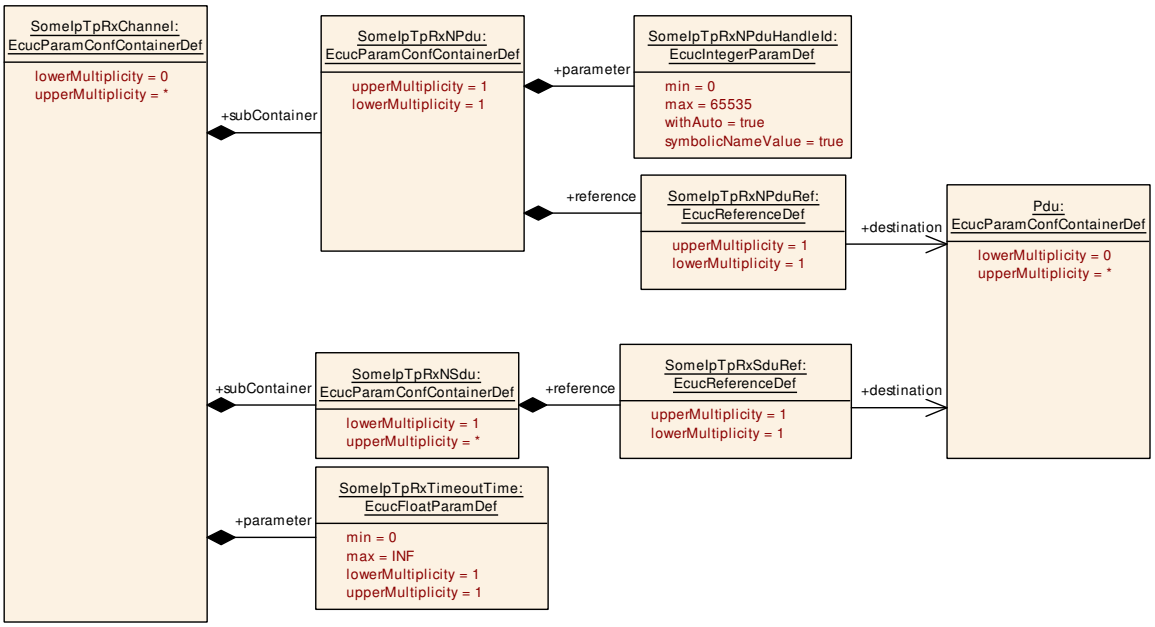

Figure 10.4

10.2.4 SomeIpTpRxNSdu

[ECUC_SomeIpTp_00008] Definition of EcucParamConfContainerDef SomeIpTpRxNSdu ⌈

Container Name	SomeIpTpRxNSdu	
Parent Container	SomeIpTpRxChannel	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	This container defines the upper layer Sdus assembled in the context of the SomeIpTpRxChannel. This PDU can forward meta data items of type SOCKET_CONNECTION_ID_16.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SomeIpTpRxSduRef	1	[ECUC_SomeIpTp_00010]

No Included Containers

⌋

[ECUC_SomeIpTp_00010] Definition of EcucReferenceDef SomeIpTpRxSduRef ⌈

Parameter Name	SomeIpTpRxSduRef	
Parent Container	SomeIpTpRxNSdu	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

10.2.5 SomeIpTpRxNPdu

[ECUC_SomeIpTp_00011] Definition of EcucParamConfContainerDef SomeIpTpRxNPdu ⌈

Container Name	SomeIpTpRxNPdu
Parent Container	SomeIpTpRxChannel
Upstream mapping defined	[7, CP_TPS_SystemTemplate]
Description	This container contains the configuration parameters of the NPdu that is received from a lower layer
Multiplicity	1

▽

△

Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SomeIpTpRxNPduHandleId	1	[ECUC_SomeIpTp_00013]
SomeIpTpRxNPduRef	1	[ECUC_SomeIpTp_00012]

No Included Containers

⌋

[ECUC_SomeIpTp_00013] Definition of EcucIntegerParamDef SomeIpTpRxNPduHandleId ⌈

Parameter Name	SomeIpTpRxNPduHandleId	
Parent Container	SomeIpTpRxNPdu	
Upstream mapping defined	–	
Description	This parameter defines the handle ID that is used by the PduR when calling SomeIp Tp_RxIndication.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_SomeIpTp_00012] Definition of EcucReferenceDef SomeIpTpRxNPduRef ⌈

Parameter Name	SomeIpTpRxNPduRef	
Parent Container	SomeIpTpRxNPdu	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	Reference to a global Pdu that is used to harmonize HandleIDs in the COM-Stack.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.6 SomeIpTpTxNSdu

[ECUC_SomeIpTp_00009] Definition of EcucParamConfContainerDef SomeIpTpTxNSdu ⌈

Container Name	SomeIpTpTxNSdu	
Parent Container	SomeIpTpTxChannel	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
Description	This container defines the upper layer Sdus to be transmitted in the context of the SomeIpTpTxChannel. This PDU can forward meta data items of type SOCKET_CONNECTION_ID_16.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
SomeIpTpTxNSduHandleId	1	[ECUC_SomeIpTp_00020]
SomeIpTpTxNSduRef	1	[ECUC_SomeIpTp_00015]

No Included Containers

⌋

[ECUC_SomeIpTp_00020] Definition of EcucIntegerParamDef SomeIpTpTxNSduHandleId ⌈

Parameter Name	SomeIpTpTxNSduHandleId	
Parent Container	SomeIpTpTxNSdu	
Upstream mapping defined	–	
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_SomeIpTp_00015] Definition of EcucReferenceDef SomeIpTpTxNSduRef ⌈

Parameter Name	SomeIpTpTxNSduRef	
Parent Container	SomeIpTpTxNSdu	
Upstream mapping defined	[7, CP_TPS_SystemTemplate]	
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 Value	Loadable	Selectable

⌋

10.2.7 SomeIpTpTxNPdu

[ECUC_SomeIpTp_00016] Definition of EcucParamConfContainerDef SomeIpTpTxNPdu ⌈

Container Name	SomeIpTpTxNPdu
Parent Container	SomeIpTpTxChannel
Upstream mapping defined	[7, CP_TPS_SystemTemplate]
Description	This container contains the configuration parameters of the segmented Tx NPdus that are transmitted to a lower layer.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
SomeIpTpTxNPduHandleId	1	[ECUC_SomeIpTp_00017]
SomeIpTpTxNPduRef	1	[ECUC_SomeIpTp_00018]

No Included Containers

⌋

[ECUC_SomeIpTp_00017] Definition of EcucIntegerParamDef SomeIpTpTxNPduHandleId ⌈

Parameter Name	SomeIpTpTxNPduHandleId
Parent Container	SomeIpTpTxNPdu
Upstream mapping defined	–
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	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_SomeIpTp_00018] Definition of EcucReferenceDef SomeIpTpTxNPduRef ⌈

Parameter Name	SomeIpTpTxNPduRef
Parent Container	SomeIpTpTxNPdu
Upstream mapping defined	[7, CP_TPS_SystemTemplate]
Description	Reference to a global Pdu that is used to harmonize HandleIDs in the COM-Stack.
Multiplicity	1

▽

△

Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.3 Published Information

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

A Not applicable requirements

[SWS_SomeIpTp_00083] Not applicable requirements

Upstream requirements: SRS_BSW_00004, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00432, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491

⌈These requirements are not applicable to this specification.⌋

B Change History

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

B.1 Change History R26-11

B.1.1 Added Specification Items in R26-11

[SWS_SomeIpTp_00083] [SWS_SomeIpTp_00098] [SWS_SomeIpTp_00099]

B.1.2 Changed Specification Items in R26-11

[ECUC_SomeIpTp_00002] [SWS_SomeIpTp_00031] [SWS_SomeIpTp_00043] [SWS_SomeIpTp_00052] [SWS_SomeIpTp_00071] [SWS_SomeIpTp_00094]

B.1.3 Deleted Specification Items in R26-11

[ECUC_SomeIpTp_00019] [SWS_SomeIpTp_00044] [SWS_SomeIpTp_00066] [SWS_SomeIpTp_00092]

B.2 Change History R25-11

B.2.1 Added Specification Items in R25-11

[ECUC_SomeIpTp_00025] [ECUC_SomeIpTp_00026] [SWS_SomeIpTp_00090] [SWS_SomeIpTp_00091] [SWS_SomeIpTp_00092] [SWS_SomeIpTp_00093] [SWS_SomeIpTp_00094] [SWS_SomeIpTp_00095] [SWS_SomeIpTp_00096] [SWS_SomeIpTp_00097] [SWS_SomeIpTp_91003]

B.2.2 Changed Specification Items in R25-11

[ECUC_SomeIpTp_00001] [ECUC_SomeIpTp_00002] [ECUC_SomeIpTp_00004] [ECUC_SomeIpTp_00005] [ECUC_SomeIpTp_00006] [ECUC_SomeIpTp_00008] [ECUC_SomeIpTp_00009] [ECUC_SomeIpTp_00010] [ECUC_SomeIpTp_00011] [ECUC_SomeIpTp_00012] [ECUC_SomeIpTp_00013] [ECUC_SomeIpTp_00015] [ECUC_SomeIpTp_00016] [ECUC_SomeIpTp_00017] [ECUC_SomeIpTp_00018] [ECUC_SomeIpTp_00019] [ECUC_SomeIpTp_00020] [ECUC_SomeIpTp_00021] [ECUC_SomeIpTp_00023] [ECUC_SomeIpTp_00024] [SWS_SomeIpTp_00017] [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_00030] [SWS_SomeIpTp_00031] [SWS_SomeIpTp_00032] [SWS_SomeIpTp_00033] [SWS_SomeIpTp_00036] [SWS_SomeIpTp_00037] [SWS_SomeIpTp_00038] [SWS_SomeIpTp_00040] [SWS_SomeIpTp_00041] [SWS_SomeIpTp_00042] [SWS_SomeIpTp_00045] [SWS_SomeIpTp_00047] [SWS_SomeIpTp_00048] [SWS_SomeIpTp_00049] [SWS_SomeIpTp_00050] [SWS_SomeIpTp_00051] [SWS_SomeIpTp_00053] [SWS_SomeIpTp_00054] [SWS_SomeIpTp_00056] [SWS_SomeIpTp_00062] [SWS_SomeIpTp_00063] [SWS_SomeIpTp_00064] [SWS_SomeIpTp_00065] [SWS_SomeIpTp_00071] [SWS_SomeIpTp_00082] [SWS_SomeIpTp_91001] [SWS_SomeIpTp_91002]

B.2.3 Deleted Specification Items in R25-11

[ECUC_SomeIpTp_00003] [SWS_SomeIpTp_00057] [SWS_SomeIpTp_00067] [SWS_SomeIpTp_00072] [SWS_SomeIpTp_00073] [SWS_SomeIpTp_00077]

B.3 Change History R24-11

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

[ECUC_SomeIpTp_00023] [SWS_SomeIpTp_00007] [SWS_SomeIpTp_00030] [SWS_SomeIpTp_00031] [SWS_SomeIpTp_00043] [SWS_SomeIpTp_00055] [SWS_SomeIpTp_00057] [SWS_SomeIpTp_00066] [SWS_SomeIpTp_00067] [SWS_SomeIpTp_00071] [SWS_SomeIpTp_00072] [SWS_SomeIpTp_00073] [SWS_SomeIpTp_00074] [SWS_SomeIpTp_00075] [SWS_SomeIpTp_00076]

B.3.3 Deleted Specification Items in R24-11

[SWS_SomeIpTp_00081]

B.4 Change History R23-11

B.4.1 Added Specification Items in R23-11

[SWS_SomeIpTp_00081] [SWS_SomeIpTp_00082]

B.4.2 Changed Specification Items in R23-11

[SWS_SomeIpTp_00017] [SWS_SomeIpTp_00052]

B.4.3 Deleted Specification Items in R23-11

none

B.5 Change History R22-11

B.5.1 Added Specification Items in R22-11

none

B.5.2 Changed Specification Items in R22-11

[SWS_SomeIpTp_00043] [SWS_SomeIpTp_00044] [SWS_SomeIpTp_00046] [SWS_SomeIpTp_00047] [SWS_SomeIpTp_00052] [SWS_SomeIpTp_00053] [SWS_SomeIpTp_00056] [SWS_SomeIpTp_00058] [SWS_SomeIpTp_00060] [SWS_SomeIpTp_00061] [SWS_SomeIpTp_00065] [SWS_SomeIpTp_00069] [SWS_SomeIpTp_91001] [SWS_SomeIpTp_91002]

B.5.3 Deleted Specification Items in R22-11

none

B.5.4 Added Constraints in R22-11

none

B.5.5 Changed Constraints in R22-11

none

B.5.6 Deleted Constraints in R22-11

none

B.5.7 Added Advisories in R22-11

none

B.5.8 Changed Advisories in R22-11

none

B.5.9 Deleted Advisories in R22-11

none