| **Document Title** | SWS_SOMEIPTransportProtocol: Complete Change Documentation 4.3.0 - 4.3.1 |
|---|---|
| **Document Owner** | AUTOSAR |
| **Document Responsibility** | AUTOSAR |
| **Document Identification No** | 695 |

| | |
|---|---|
| **Document Status** | Final |
| **Part of AUTOSAR Standard** | Classic Platform |
| **Part of Standard Release** | 4.3.1 |

# Table of Contents

# 1 SWS_SOMEIPTransportProtocol

## 1.1 Specification Item ECUC_SomeIpTp_00017

**Trace References:**

none

**Content:**

| | | | |
|---|---|---|---|
| Name | SomeIpTpTxNPduHandleIdSomeIpTpTxNPdu.SomeIpTpTxNPduHandleId | | |
| Parent Container | SomeIpTpTxNPdu | | |
| Description | This parameter defines the handle ID that is used by PduR when calling SomeIpTp_Trigger Transmit. | | |
| Multiplicity | 0..1 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 | | |

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connec-

tion does not need to be terminated.

SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRx-TimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

Multiplicity of SomeIpTpTxNPduHandleId should be 1.

The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

**Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.2 Specification Item ECUC_SomeIpTp_00023

**Trace References:**

none

**Content:**

| | | | |
|---|---|---|---|
| Name | SomeIpTpRxTimeoutTimeSomeIpTpChannel.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 = SomeIpTpNPdu SeparationTime + 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 | | |

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

Multiplicity of SomeIpTpTxNPduHandleId should be 1.

The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

**Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.3 Specification Item SWS_SomeIpTp_00022

**Trace References:**

RS_SOMEIP_00010

**Content:**

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.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

  Multiplicity of SomeIpTpTxNPduHandleId should be 1.

  The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

  **Agreed solution:**

  - The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
  - In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".

- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".

–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.4 Specification Item SWS_SomeIpTp_00028

**Trace References:**

RS_SOMEIP_00010

**Content:**

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

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

Multiplicity of SomeIpTpTxNPduHandleId should be 1.

The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

**Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.5 Specification Item SWS_SomeIpTp_00036

**Trace References:**

RS_SOMEIP_00010

**Content:**

The SOME/IP TP module shall store the amount number of Payload bytes for every PDU ID separately which has been passed to the upper layer.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

  Multiplicity of SomeIpTpTxNPduHandleId should be 1.

  The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

  **Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.6 Specification Item SWS_SomeIpTp_00037

**Trace References:**

RS_SOMEIP_00010

**Content:**

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.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

  Multiplicity of SomeIpTpTxNPduHandleId should be 1.

  The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

  **Agreed solution:**

  - The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
  - In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
  - ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.7 Specification Item SWS_SomeIpTp_00038

**Trace References:**

RS_SOMEIP_00010

**Content:**

The SOME/IP TP module shall store the amount number of Payload bytes for every PDU ID separately which has been passed to the upper layer.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the

next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

Multiplicity of SomeIpTpTxNPduHandleId should be 1.

The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

**Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.8 Specification Item SWS_SomeIpTp_00045

**Trace References:**

RS_SOMEIP_00010

**Content:**

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_SomeIp Tp_00083. 00054.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_MESSAGE_TYPE.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

  Multiplicity of SomeIpTpTxNPduHandleId should be 1.

The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

**Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".

–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.9 Specification Item SWS_SomeIpTp_00060

**Trace References:**

SRS_BSW_00384

**Content:**

| API function | Description |
|---|---|
| Det_ReportRuntimeError | Service to report runtime errors. If a callout has been configured then this callout shall be called. |

| API function | Description |
|---|---|
| PduR_SomeIpTpCopyRxData | 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 data buffer is written to the position indicated by bufferSizePtr. |
| PduR_SomeIpTpCopyTxData | This function is called to acquire the transmit data of an I-PDU segment (N-PDU).<br>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->TxTpData Cnt. The size of the remaining data is written to the position indicated by availableDataPtr. |
| PduR_SomeIpTpRxIndication | Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not. |
| PduR_SomeIpTpStartOfReception | 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 | Requests transmission of a PDU. |
| PduR_SomeIpTpTxConfirmation | This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not. |

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #77935: [PduR] Misleading description of CopyRxData

  **Problem description:**

  ---

  Name: Martin Schlodder
  Role: Member of WP-A2

  ---

  Description/Motivation:

  The description of the CopyRxData API says: "The size of the remaining data is written to the position indicated by bufferSizePtr."

  This text seems to have been copied from the CopyTxData call, where it is correct. CopyRxData should talk about "remaining buffer", not "remaining data".

  **Agreed solution:**

  In the description of the API PduR_<User:LoTp>CopyRxData (SWS_PduR_00512), replace "remaining data" by "remaining buffer".

  **BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 1 | 1 |

## 1.10 Specification Item SWS_SomeIpTp_00062

**Trace References:**

RS_SOMEIP_00010

**Content:**

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_SomeIp Tp_00083. 00054.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_INCONSISTENT_HEADER.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY

is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRx-TimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

Multiplicity of SomeIpTpTxNPduHandleId should be 1.

The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

**Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.11 Specification Item SWS_SomeIpTp_00063

**Trace References:**

RS_SOMEIP_00010, PRS_SOMEIP_00749

**Content:**

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_SomeIp Tp_00083. 00054.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_ASSEMBLY_INTERRUPT.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

  Multiplicity of SomeIpTpTxNPduHandleId should be 1.

  The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

  **Agreed solution:**

- The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
- In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
- ECUC_SomeIpTp_00023: Change to "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."
- SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
- SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |

## 1.12 Specification Item SWS_SomeIpTp_00064

**Trace References:**

RS_SOMEIP_00010, PRS_SOMEIP_00749

**Content:**

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_SomeIp Tp_00083. 00054.
- The API Det_ReportRuntimeError() shall be called with the runtime error code SOMEIPTP_E_INCONSISTENT_SEQUENCE.

**RfCs affecting this spec item between releases 4.3.0 and 4.3.1:**

- RfC #76751: [SomeIpTp] Several findings

  **Problem description:**

  When we started activities for implementation of this protocol, we detected the following shortcomings of the specification:

  The SomeIpTpRxTimeoutTime is defined per channel for the whole transmission. It will be absurdly long for short transmission, if there is at least on very large N-SDU configured. It would be better to use it to define the timeout between two consecutive N-PDUs of one N-SDU reception.

  SWS_SomeIpTp_00039 is not helpful. Storing one frame only helps until the next frame is received, and if the sender is slower than the receiver and the buffer too small (PduR), there will be an abort anyways. Suggestion: Just pass on everything that was received directly to the upper layer, and stop processing if this fails.

  SWS_SomeIpTp_00022 should only deny a new connection, the old connection does not need to be terminated.

  SWS_SomeIpTp_00024 is not helpful in case there is a delay in a routed message. It would make more sense to just try CopyTxData as long as BUFREQ_E_BUSY is returned, or no timeout (suggestion: SomeIpTpTxTimeoutTime < SomeIpTpRxTimeoutTime) occurred. See also SWS_SomeIpTp_00025 and others.

  Multiplicity of SomeIpTpTxNPduHandleId should be 1.

  The example in section 7.1.5 should be changed such that it is obvious that the N-SDU can have any length at run time, not just multiples of 16 (or 8).

  **Agreed solution:**

  - The example in section 7.1.5 shall be changed such that it uses an overall length of 5579.
  - In SWS_SomeIpTp_00028, SWS_SomeIpTp_00036, and SWS_SomeIpTp_00038, replace "the amount of" by "the number of".
  - ECUC_SomeIpTp_00023: Change to "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."
  - SWS_SomeIpTp_00037: Add "restart the SomeIpTpRxTimeoutTime and" in the last sentence before "continue with".
  - SWS_SomeIpTp_00022, add "In The API PduR_SomeIpTpTxConfirmation()with

result set to E_NOT_OK shall be called." as third item.
- Set multiplicity of SomeIpTpTxNPduHandleId (ECUC_SomeIpTp_00017) to 1.
- Replace references to SWS_SomeIpTp_00083 by SWS_SomeIpTp_00054.
- In SWS_SomeIpTp_00045, insert the following text after "If the API SomeIpTp_RxIndication() is called": "and a session is currently active".
–Last change on issue 76751 comment 10–

**BW-C-Level:**

| Application | Specification | Bus |
|---|---|---|
| 1 | 4 | 1 |