

<b>Document Title</b>	Specification of CAN Transport Layer
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	014
<b>Document Classification</b>	Standard

<b>Document Version</b>	5.2.0
<b>Document Status</b>	Final
<b>Part of Release</b>	4.1
<b>Revision</b>	3

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
31.03.2014	5.2.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Revised padding behaviour.</li> <li>Clarified relation between CanTpMainFunctionPeriod and other timers.</li> <li>Revised CanTp_RxIndication() prototype.</li> <li>Extended parameter CanTpTc for receive cancellation.</li> </ul>
31.10.2013	5.1.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Replace NTFRSLT_OK/NTFRSLT_&lt;other&gt; E_OK/E_NOT_OK</li> <li>Handling of unexpected arrival of N-PDU table clarification</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>
28.02.2013	5.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Error handling has been improved</li> <li>PostBuild concept has been refined</li> <li>Introduction of HDV support</li> <li>Clarifications of buffer handling</li> </ul>
01.12.2011	4.0.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>CanTp does not report production errors anymore</li> <li>Metamodel structure changed</li> <li>Harmonization with the new buffer concept</li> <li>Change the BlockSize to be statically configurable instead a maximum value</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
22.10.2010	3.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Corrections and improvement in errors description;</li> <li>• API services correction;</li> <li>• Clarifications in relation with buffer handling</li> <li>• Updated table in Ch.6 for half and full duplex support</li> </ul>
07.12.2009	3.0.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Added Mixed Addressing Mode</li> <li>• CanTp supports Full Duplex Mode</li> <li>• New buffering concept</li> <li>• Added possibility to change CanTp parameters</li> <li>• Legal disclaimer revised</li> </ul>
03.12.2007	2.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Addition of transmit cancellation feature</li> <li>• DataLength check only for too small DLC (CanTp220)</li> <li>• Restriction on mapping of N-Pdu (SWS_CanTp_00248)</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>
24.01.2007	2.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• “Advice for users” revised</li> <li>• “Revision Information” added</li> </ul>
04.12.2006	2.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Clarification and correction of error management: list of production/development error and behavior in case of error</li> <li>• Addition of SWS_CanTp_00166 and SWS_CanTp_00167 to avoid blocking situation in case of no buffer provided by upper layer</li> <li>• Remove of CanTpRxWftMax of container CanTpTxNSdu</li> <li>• 1 parameter added for the call of Det_ReportError</li> <li>• Add header files inclusions</li> <li>• Addition of CanTpNSa container in configuration chapter</li> <li>• Legal disclaimer revised</li> </ul>
27.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template.
21.06.2005	1.0.0	AUTOSAR Administration	Initial Release

**Disclaimer**

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

**Advice for users**

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

## Table of Contents

1	Introduction and functional overview .....	7
2	Acronyms and abbreviations .....	10
3	Related documentation.....	13
3.1	Input documents .....	13
3.2	Related standards and norms .....	14
3.3	Related specification .....	14
4	Constraints and assumptions .....	15
4.1	Limitations .....	15
4.2	Applicability in automotive domain .....	15
5	Dependencies on other modules.....	16
5.1	AUTOSAR architecture basic concepts.....	16
5.1.1	CAN Transport Layer connection(s).....	16
5.1.2	CAN Transport Layer interactions.....	16
5.1.3	Processing mode .....	17
5.1.4	Data consistency.....	17
5.1.5	Static configuration .....	17
5.1.6	PDU Router services .....	18
5.1.7	CAN Interface services .....	18
5.2	File structure.....	18
5.2.1	Code file structure.....	19
5.2.2	Header file structure.....	19
5.2.3	Version check .....	20
5.2.4	Design Rules.....	20
6	Requirements traceability .....	21
7	Functional specification .....	29
7.1	Services provided to upper layer .....	29
7.1.1	Initialization and shutdown .....	29
7.1.2	Transmit request.....	31
7.1.3	Transmit cancellation .....	31
7.2	Services provided to the lower layer.....	32
7.2.1	Transmit confirmation .....	32
7.2.2	Reception indication.....	32
7.3	Internal behavior.....	33
7.3.1	N-SDU Reception .....	33
7.3.2	N-SDU Transmission .....	37
7.3.3	Buffer strategy.....	39
7.3.4	Protocol parameter setting services.....	42
7.3.5	Tx and Rx data flow .....	42
7.3.6	Relationship between CAN NSduld and CAN LSduld .....	43
7.3.7	Concurrent connection.....	45
7.3.8	N-PDU padding.....	46
7.3.9	Handling of unexpected N-PDU arrival .....	47
7.4	Error classification .....	49
7.5	Error detection.....	51

7.6	Error notification .....	51
7.7	AUTOSAR debugging concept .....	52
8	API specification .....	53
8.1	Imported types .....	53
8.2	Type definitions .....	53
8.2.1	CanTp_ConfigType .....	53
8.3	Function definitions .....	54
8.3.1	CanTp_Init .....	54
8.3.2	CanTp_GetVersionInfo .....	55
8.3.3	CanTp_Shutdown .....	55
8.3.4	CanTp_Transmit .....	56
8.3.5	CanTp_CancelTransmit .....	58
8.3.6	CanTp_CancelReceive .....	59
8.3.7	CanTp_ChangeParameter .....	60
8.3.8	CanTp_ReadParameter .....	61
8.3.9	Main Function .....	61
8.4	Call-back notifications .....	62
8.4.1	CanTp_RxIndication .....	62
8.4.2	CanTp_TxConfirmation .....	63
8.5	Expected Interfaces .....	63
8.5.1	Mandatory Interfaces .....	63
8.5.2	Optional Interfaces .....	64
9	Sequence diagrams .....	65
9.1	SF N-SDU received and no buffer available .....	65
9.1.1	Assumptions .....	65
9.1.2	Sequence diagram .....	65
9.1.3	Transition description .....	66
9.2	Successful SF N-PDU reception .....	67
9.2.1	Assumptions .....	67
9.2.2	Sequence diagram .....	67
9.2.3	Transition description .....	68
9.3	Transmit request of SF N-SDU .....	68
9.3.1	Assumptions .....	68
9.3.2	Sequence diagram .....	69
9.3.3	Transition description .....	70
9.4	Transmit request of larger N-SDU .....	71
9.4.1	Assumptions .....	71
9.4.2	Sequence diagram .....	72
9.4.3	Transition description .....	73
9.5	Large N-SDU Reception .....	74
9.5.1	Assumptions .....	74
9.5.2	Sequence diagram .....	75
9.5.3	Transition description .....	76
10	Configuration specification .....	77
10.1	How to read this chapter .....	77
10.2	Containers and configuration parameters .....	78
10.2.1	Variants .....	78
10.2.2	CanTp .....	78

10.2.3	CanTpConfig .....	78
10.2.4	CanTpGeneral .....	79
10.2.5	CanTpChannel .....	81
10.2.6	CanTpRxNSdu .....	81
10.2.7	CanTpRxNPdu .....	85
10.2.8	CanTpTxFcNPdu .....	86
10.2.9	CanTpTxNSdu .....	86
10.2.10	CanTpTxNPdu .....	89
10.2.11	CanTpRxFcNPdu .....	90
10.2.12	CanTpNTa .....	91
10.2.13	CanTpNSa .....	91
10.2.14	CanTpNAe .....	92
10.3	Published Information .....	93
11	Not applicable requirements .....	94

## 1 Introduction and functional overview

This specification defines the functionality, API and the configuration of the AUTOSAR Basic Software module CAN Transport Layer (CanTp).

CanTp is the module between the PDU Router and the CAN Interface module (see Figure 1). The main purpose of the CAN TP module is to segment and reassemble CAN I-PDUs longer than 8 bytes.

The PDU Router deploys AUTOSAR COM and DCM I-PDUs onto different communication protocols. The routing through a network system type (e.g. CAN, LIN and FlexRay) depends on the I-PDU identifier. The PDU Router also determines if a transport protocol has to be used or not. Lastly, this module carries out gateway functionality, when there is no rate conversion.

CAN Interface (CanIf) provides equal mechanisms to access a CAN bus channel regardless of its location ( $\mu$ C internal/external). From the location of CAN controllers (on chip / onboard), it extracts the ECU hardware layout and the number of CAN drivers. Because CanTp only handles transport protocol frames (i.e. SF, FF, CF and FC PDUs), depending on the N-PDU ID, the CAN Interface has to forward an I-PDU to CanTp or PduR.

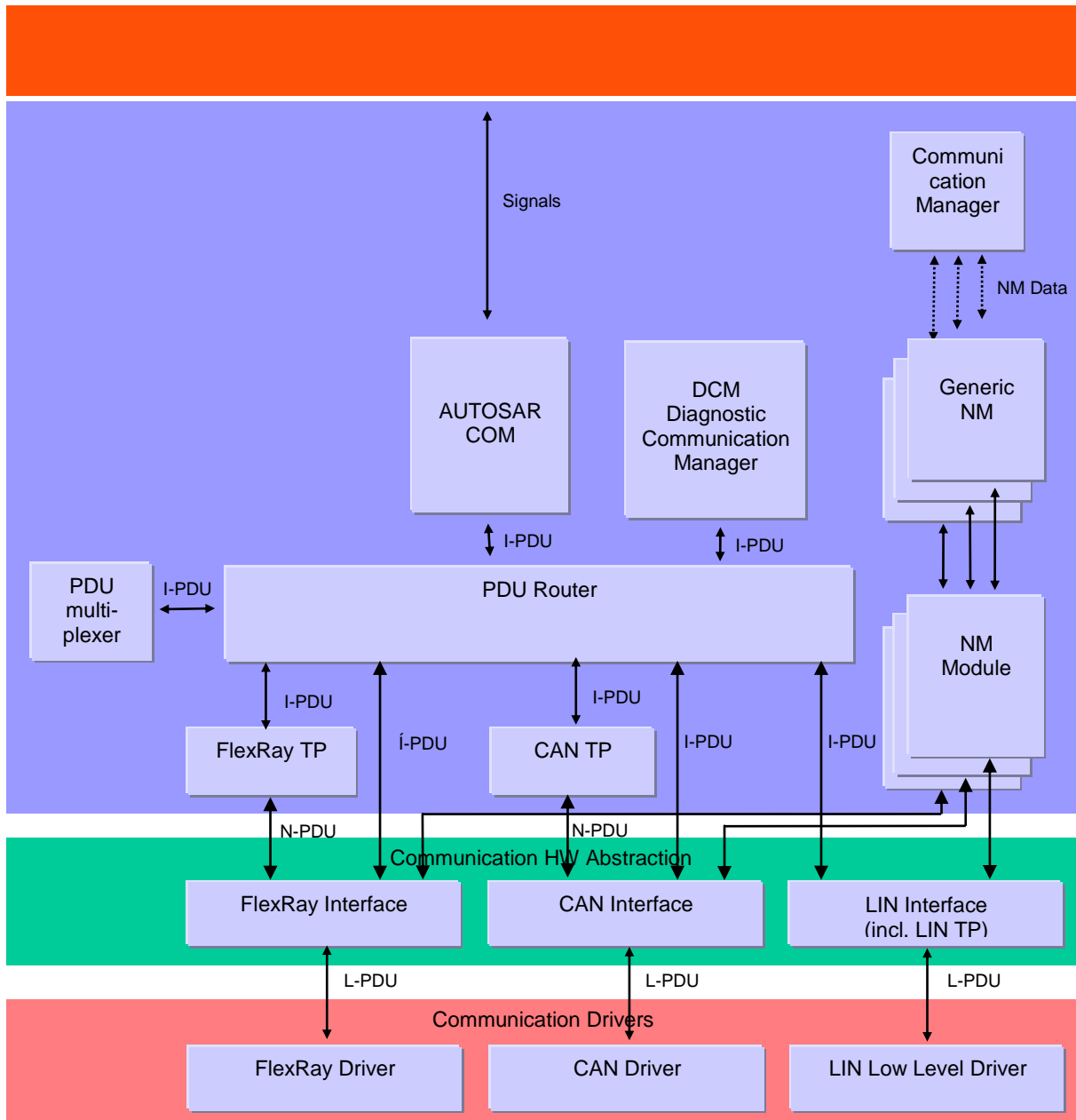


Figure 1 : AUTOSAR Communication Stack

According to AUTOSAR basic software architecture, CanTp provides services for:

- Segmentation of data in transmit direction;
- Reassembling of data in receive direction;
- Control of data flow;
- Detection of errors in segmentation sessions.
- Transmit cancellation
- Receive cancellation

It is an AUTOSAR decision to base basic software module specifications on existing standards, thus this AUTOSAR CAN Transport Layer specification is based on the international standard ISO 15765, which is the most used standard in the automotive domain.

ISO 15765 (containing four sections) describes two applicable CAN Transport Layer specifications: ISO 15765-2 for OEM enhanced diagnostics [14] and ISO 15765-4 for



OBD diagnostics [16]. Concerning the transport layer, ISO 15765-4 (the section of ISO 15765 which also covers the data link layer and physical layer) is in accordance with ISO 15765-2 with some restrictions/additions. In order that there is no incompatibility problem between ISO 15765-2 and ISO 15765-4, differences will be solved by the CAN Transport Layer configuration.

Although CAN transport protocol is mainly used for vehicle diagnostic systems, it has also been developed to deal with requirements from other CAN based systems requiring a transport layer protocol.

## 2 Acronyms and abbreviations

The prefix notation used in this document, is as follows:

<b>Prefix:</b>	<b>Description:</b>
I-	Relative to AUTOSAR COM Interaction Layer
L-	Relative to the CAN Interface module which is equivalent to the Logical Link Control (the upper part of the Data Link Layer – the lower part is called Media Access Control)
N-	Relative to the CAN Transport Layer which is equivalent to the OSI Network Layer.

All acronyms and abbreviations, which are specific to the CAN Transport Layer and are therefore not contained in the AUTOSAR glossary, are described in the following:

<b>Acronym:</b>	<b>Description:</b>
CAN L-SDU	This is the SDU of the CAN Interface module. It is similar to CAN N-PDU but from the CAN Interface module point of view.
CAN LSduId	This is the unique identifier of a SDU within the CAN Interface. It is used for referencing L-SDU's routing properties. Consequently, in order to interact with the CAN Interface through its API, an upper layer uses CAN LSduId to refer to a CAN L-SDU Info Structure.
CAN N-PDU	This is the PDU of the CAN Transport Layer. It contains a unique identifier, data length and data (protocol control information plus the whole N-SDU or a part of it).
CAN N-SDU	This is the SDU of the CAN Transport Layer. In the AUTOSAR architecture, it is a set of data coming from the PDU Router.
CAN N-SDU Info Structure	This is a CAN Transport Layer internal constant structure that contains specific CAN Transport Layer information to process transmission, reception, segmentation and reassembly of the related CAN N-SDU.
CAN NSduId	Unique SDU identifier within the CAN Transport Layer. It is used to reference N-SDU's routing properties. Consequently, to interact with the CAN Transport Layer via its API, an upper layer uses CAN NSduId to refer to a CAN N-SDU Info Structure.
I-PDU	This is the PDU of the AUTOSAR COM module.
PDU	In layered systems, it refers to a data unit that is specified in the protocol of a given layer. This contains user data of that layer (SDU) plus possible protocol control information. Furthermore, the PDU of layer X is the SDU of its lower layer X-1 (i.e. (X)-PDU = (X-1)-SDU).
PduInfoType	This type refers to a structure used to store basic information to process the transmission/reception of a PDU (or a SDU), namely a pointer to its payload in RAM and the corresponding length (in bytes).
SDU	In layered systems, this refers to a set of data that is sent by a user of the services of a given layer, and is transmitted to a peer service user, whilst remaining semantically unchanged.

<b>Abbreviation:</b>	<b>Description:</b>
BS	Block Size
Can	CAN Driver module
CAN CF	CAN Consecutive Frame N-PDU
CAN FC	CAN Flow Control N-PDU
CAN FF	CAN First Frame N-PDU
CAN SF	CAN Single Frame N-PDU

<b>Abbreviation:</b>	<b>Description:</b>
CanIf	CAN Interface
CanTp	CAN Transport Layer
CanTrcv	CAN Transceiver module
CF	See "CAN CF"
Com	AUTOSAR COM module
Dcm	Diagnostic Communication Manager module
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DLC	Data Length Code (part of CAN PDU that describes the SDU length)
FC	See "CAN FC"
FF	See "CAN FF"
FIM	Function Inhibition Manager
Mtype	Message Type (possible value: diagnostics, remote diagnostics)
N_AI	Network Address Information (see ISO 15765-2).
N_Ar	Time for transmission of the CAN frame (any N-PDU) on the receiver side (see ISO 15765-2).
N_As	Time for transmission of the CAN frame (any N-PDU) on the sender side (see ISO 15765-2).
N_Br	Time until transmission of the next flow control N-PDU (see ISO 15765-2).
N_Bs	Time until reception of the next flow control N-PDU (see ISO 15765-2).
N_Cr	Time until reception of the next consecutive frame N-PDU (see ISO 15765-2).
N-Cs	Time until transmission of the next consecutive frame N-PDU (see ISO 15765-2).
N_Data	Data information of the transport layer
N_PCI	Protocol Control Information of the transport layer
N_SA	Network Source Address (see ISO 15765-2).
N_TA	Network Target Address (see ISO 15765-2). It might already contain the N_TAtype(physical/function) in case of ExtendedAddressing.
N_TAtype	Network Target Address type (see ISO 15765-2).
OBD	On-Board Diagnostic
PDU	Protocol Data Unit
PduR	PDU Router
SDU	Service Data Unit
FS	Flow Status

The following table contains some of the concepts, which are useful in this work:

<b>Definitions:</b>	<b>Description:</b>
Development Error Tracer	The Development Error Tracer is merely a support to SW development and integration and is <u>not</u> contained in the production code. The API is defined, but the functionality can be chosen and implemented by the developer according to his specific needs.
Diagnostic Event Manager	The Diagnostic Event Manager is a standard AUTOSAR module which is available in the production code and whose functionality is specified in the AUTOSAR project.
Extended addressing format	A unique CAN identifier is assigned to each combination of N_SA and Mtype. A unique address is filed to each combination of N_TA and N_TAtype in the first data byte of the CAN frame data field. N_PCI and N_Data are filed in the remaining bytes of the CAN frame data field.
Full-duplex	Point-to-point communication between two nodes is possible in both directions at any one time.

<b>Definitions:</b>	<b>Description:</b>
Function Inhibition Manager	The Function Inhibition Manager (FIM) stands for the evaluation and assignment of events to the required actions for Software Components (e.g. inhibition of specific “monitoring functions”). The DEM informs and updates the Function Inhibition Manager (FIM) upon changes of the event status in order to stop or release functional entities according to assigned dependencies. An interface to the functional entities is defined and supported by the Mode Manager. The FIM is not part of the DEM.
Functional addressing	<p>In the transport layer, functional addressing refers to N-SDU, of which parameter N_TAtype (which is an extension to the N_TA parameter [14] used to encode the communication model) has the value <i>functional</i>.</p> <p>This means the N-SDU is used in 1 to n communications. Thus with the CAN protocol, functional addressing will only be supported for Single Frame communication.</p> <p>In terms of application, functional addressing is used by the external (or internal) tester if it does not know the physical address of an ECU that should respond to a service request or if the functionality of the ECU is implemented as a distributed server over several ECUs. When functional addressing is used, the communication is a communication broadcast from the external tester to one or more ECUs (1 to n communication).</p> <p>Use cases are (for example) broadcasting messages, such as “ECUReset” or “CommunicationControl”</p> <p>OBd communication will always be performed as part of functional addressing.</p>
Half-duplex	Point-to-point communication between two nodes is only possible in one direction at a time.
Mixed addressing format	A unique CAN identifier is assigned to each combination of N_SA, N_TA, N_TAtype. N_AE is placed in the first data byte of the CAN frame data field. N_PCI and N_Data are placed in the remaining bytes of the CAN frame data field.
Multiple connection	The CAN Transport Layer should manage several transport protocol communication sessions at a time.
Normal addressing format	A unique CAN identifier is assigned to each combination of N_SA, N_TA, N_TAtype and Mtype. N_PCI and N_Data are filed in the CAN frame data field.
Physical addressing	<p>In the transport layer, physical addressing refers to N-SDU, of which parameter N_TAtype (which is an extension of the N_TA parameter [14] used to encode the communication model) has the value <i>physical</i>.</p> <p>This means the N-SDU is used in 1 to 1 communication, thus physical addressing will be supported for all types of network layer messages.</p> <p>In terms of application, physical addressing is used by the external (or internal) tester if it knows the physical address of an ECU that should respond to a service request. When physical addressing is used, a point to point communication takes place (1 to 1 communication).</p> <p>Use cases are (for example) messages, such as “ReadDataByIdentifier” or “InputOutputControlByIdentifier”</p>
Single connection	The CAN Transport Layer will only manage one transport protocol communication session at a time.
Connection channel	The CAN Transport Layer is handling resources used by multiple connections in order to save RAM. When a connection becomes active, the channel that is used by this connection will be unavailable for other connections.
Connection	A transport protocol session, either is a transmission or a reception session on a N-SDU.

### 3 Related documentation

#### 3.1 Input documents

- [1] List of Basic Software Modules,  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture,  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules,  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Specification of ECU Configuration,  
AUTOSAR\_TPS\_ECUConfiguration.pdf
- [5] Glossary  
AUTOSAR\_TR\_Glossary.pdf
- [6] Requirements on CAN  
AUTOSAR\_SRS\_CAN.pdf
- [7] Specification of CAN Interface  
AUTOSAR\_SWS\_CANInterface.pdf
- [8] API Specification of Development Error Tracer  
AUTOSAR\_SWS\_DevelopmentErrorTracer.pdf
- [9] Specification of Function Inhibition Manager  
AUTOSAR\_SWS\_FunctionInhibitionManager.pdf
- [10] Specification of PDU Router  
AUTOSAR\_SWS\_PDURouter.pdf

- [11] Specification of Diagnostic Event Manager  
AUTOSAR\_SWS\_DiagnosticEventManager.pdf
- [12] Basic Software Module Description Template,  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [13] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### **3.2 Related standards and norms**

- [14] ISO 15765-2 (2004-10-12), Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part2: Network layer services
- [15] ISO 15765-3 (2004-10-06), Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part3: Implementation of diagnostic services
- [16] ISO 15765-4 (2005-01-04), Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part4: Requirements for emissions-related systems

### **3.3 Related specification**

AUTOSAR provides a General Specification on Basic Software modules [13] (SWS BSW General), which is also valid for CAN Transport Layer.

Thus, the specification SWS BSW General shall be considered as additional and required specification for CAN Transport Layer.

## 4 Constraints and assumptions

### 4.1 Limitations

The AUTOSAR architecture defines communication system specific transport layers (CanTp, LinTp including LinIf, FlexRayTp). Thus the CAN Transport Layer only covers CAN transport protocol specifics.

The CAN Transport Layer has an interface to a single underlying CAN Interface Layer and a single upper PDU Router module.

According to the AUTOSAR release plan, this CAN Transport Layer specification has the following restriction:

- CAN Transport Layer runs only in an event triggered mode

This CAN Transport Layer implementation supports half and full-duplex communication; support for full-duplex communication is configurable on channel base (see Chapter 10).

### 4.2 Applicability in automotive domain

The CAN Transport Layer can be used for all domains whenever the CAN communication system is connected to the appropriate ECU.

## 5 Dependencies on other modules

This section sets out relations between the CanTp and other AUTOSAR basic software modules. It contains short descriptions of some AUTOSAR basic concepts, configuration information and services, which are required by the CanTp from other modules.

### 5.1 AUTOSAR architecture basic concepts

#### 5.1.1 CAN Transport Layer connection(s)

In the AUTOSAR architecture final release, transport protocol facilities will be used to transport both diagnostic (e.g. OBD and UDS protocols) and AUTOSAR COM I-PDUs. Therefore, the CanTp module is able to deal with multiple connections simultaneously (i.e. multiple segmentation sessions in parallel).

The maximum number of simultaneous connections is statically configured. This configuration has an important impact on complexity and resource consumption (CPU, ROM and RAM) of the code generated, because resources (e.g. Rx and Tx state machines, variables used to work on N-PCI data and so on) have to be reserved for each simultaneous access.

To allow the user to choose which I-PDUs could be received (or sent) simultaneously, each N-SDU identifier will be internally routed through a configured CanTp “connection channel”. Since a “connection channel” is not accessible externally, all necessary information (see chapter 10.2) to transfer an N-SDU will be linked to the N-SDU identifier (e.g. “connection channel” number, timeouts, addressing format, and so on).

Depending on the MetaDataLength, an N-SDU acts either as a specific connection with defined N\_AI, or as a generic connection, where the N\_TA, N\_SA, and N\_AE vary at runtime, while N\_TAtype, MType, and the addressing format are statically defined.

#### 5.1.2 CAN Transport Layer interactions

The figure below shows the interactions between CanTp, PduR and CanIf modules.

The CanTp’s upper interface offers the PduR module global access, to transmit and receive data. This access is achieved by CAN N-SDU identifier (CAN NSduId). CAN NSduId refers to a constant data structure which consists of attributes describing CAN N-SDU. Each CAN N-SDU specific data structure may contain attributes such as: type of N-SDU (Tx or Rx), its addressing format, L-SDU identifier of this message or other attributes that are useful for implementation.



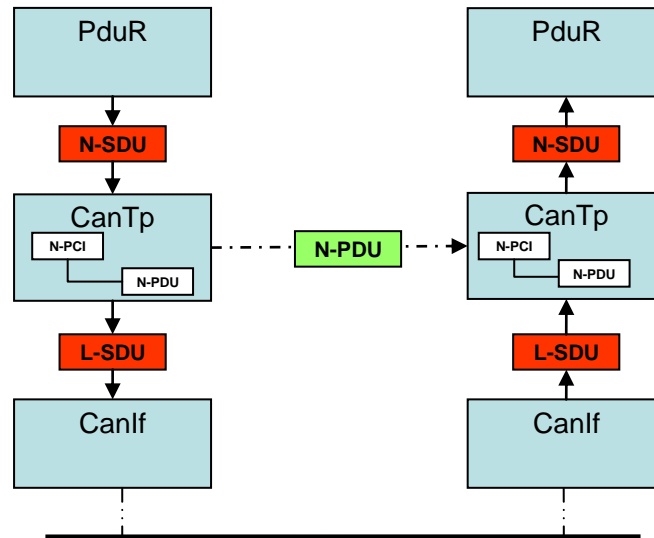


Figure 2: CAN Transport Layer interactions

### 5.1.3 Processing mode

The AUTOSAR communication stack supports both polling and event triggering mode. Therefore, each communication layer can receive information from its lower layer and propagate information to its upper layer by different mechanisms. In the case of the CAN Transport Layer, only the event triggering mode is supported.

### 5.1.4 Data consistency

To optimize the communication stack, AUTOSAR limits the CAN Transport Layer buffering capacity. Therefore, the CanTp copies N-SDU payload directly from the upper layer (DCM, COM or PDU Router – in the case of 1:1 TP routing) to the CAN driver and vice-versa. Thus to guarantee data consistency, the upper layer will observe the following rules:

- At transmission time, the N-SDU data payload will remain unchanged, from transmit request until transmit confirmation has been received
- At reception time, the N-SDU data access will be locked, from start of reception until the reception indication has been received.

### 5.1.5 Static configuration

At runtime the CAN Transport module must have all information required to manage transport connection. Therefore, the following properties should be statically configured:

- Number of CAN N-SDU
- Unique identifier of each CAN N-SDU
- Communication direction of each CAN N-SDU (Tx or Rx)
- Communication type on each channel – half or full-duplex

- Addressing format of each connection (normal, extended, mixed 11bit, normal fixed, or mixed 29 bit) and, depending on the addressing format, additionally:
  - Normal: none
  - Extended: N\_TA
  - Mixed 11 bit: N\_AE
  - Normal fixed: N\_TA, N\_SA
  - Mixed 29 bit: N\_TA, N\_SA, N\_AE

The static addressing information may be omitted for generic connections that use N-SDUs with MetaData.

- Addressing format of each connection (normal, extended or mixed) and, in the case of extended addressing format, the N\_TA value or in case of mixed addressing format the N\_AE value.
- Associated CAN L-SDU identifier of each CAN N-SDU identifier and if necessary (multiple frame segmentation session) the CAN L-SDU identifier used to transmit the CAN FC N-PDU

The configuration of the CAN Transport Layer can be performed during compilation or post-build (See chapter 10).

### 5.1.6 PDU Router services

The CAN Transport Layer uses callback functions of the PDU Router to copy transmit data and to confirm transmission, to initiate reception, copy received data and to indicate reception of a message:

- *PduR\_CanTpRxIndication*
- *PduR\_CanTpStartOfReception*
- *PduR\_CanTpCopyRxData*
- *PduR\_CanTpCopyTxData*
- *PduR\_CanTpTxConfirmation*

For more information about these functions, refer to the PDU Router module specification [10].

### 5.1.7 CAN Interface services

The CAN Transport Layer uses the following services of the CAN Interface to transmit CAN N-PDUs:

- *CanIf\_Transmit*

For more information about this function, refer to the CAN Interface module specification [7].

## 5.2 File structure

**5.2.1 Code file structure**

For details refer to the chapter 5.1.6 “Code file structure” in *SWS\_BSWGeneral*.

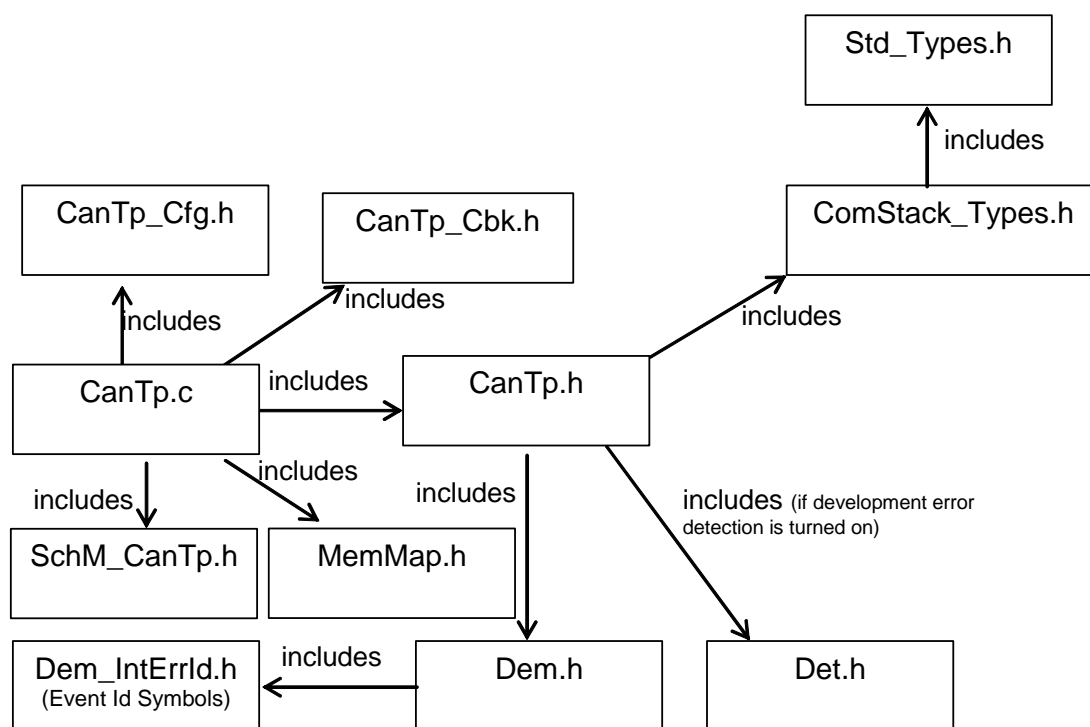
**5.2.2 Header file structure**

AUTOSAR specifies that an ECU can be created from modules provided as object code, source code (generated or not) and even mixed.

The decision to provide a module as object code or source code is based on a compromise between IP protection, test coverage, code efficiency and configurability at system generation time. Thus depending on the configurability requirements of the OEM, suppliers may deliver the CanTp module as object code, generated code or source code.

The header file structure defined in this section allows the separation of platform, compiler and implementation specific definitions and declarations from general definitions, as well as the separation of source code and configuration.

**[SWS\_CanTp\_00156]** [The CanTp module shall implement the file structure as shown in Figure 3.



**Figure 3: File Structure**

] (BSW412, SRS\_BSW\_00435, SRS\_BSW\_00436, SRS\_BSW\_00346, SRS\_BSW\_00158, SRS\_BSW\_00370, SRS\_BSW\_00301)

**[SWS\_CanTp\_00264]** [File CanTp.h shall include ComStack\_Types.h and Det.h (optional, if development error detection is configured as ON). ] ( )

**[SWS\_CanTp\_00001]** [CanTp\_Cfg.h shall define constant and customizable data for module configuration at pre-compile time. ] (SRS\_BSW\_00345, BSW381, SRS\_BSW\_00158)

**[SWS\_CanTp\_00221]** [CanTp.c shall include CanTp\_Cfg.h. ] ( )

**[SWS\_CanTp\_00160]** [References to c-configuration parameters (link time and post-build time) will be placed in a separate h-file. The h-file should be the same as pre-compilation parameters. ] ( )

### 5.2.3 Version check

**[SWS\_CanTp\_00267]** [Version number macros can be used for checking and reading out the software version of a software module, during compile-time and run-time. ] ( )

### 5.2.4 Design Rules

For details refer to the chapters 7.1.4, 7.1.8, 7.1.9 in *SWS\_BSWGeneral*.

## 6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_CanTp_00027
-	-	SWS_CanTp_00064
-	-	SWS_CanTp_00067
-	-	SWS_CanTp_00075
-	-	SWS_CanTp_00076
-	-	SWS_CanTp_00078
-	-	SWS_CanTp_00079
-	-	SWS_CanTp_00080
-	-	SWS_CanTp_00081
-	-	SWS_CanTp_00082
-	-	SWS_CanTp_00084
-	-	SWS_CanTp_00086
-	-	SWS_CanTp_00087
-	-	SWS_CanTp_00089
-	-	SWS_CanTp_00090
-	-	SWS_CanTp_00091
-	-	SWS_CanTp_00092
-	-	SWS_CanTp_00093
-	-	SWS_CanTp_00094
-	-	SWS_CanTp_00095
-	-	SWS_CanTp_00111
-	-	SWS_CanTp_00115
-	-	SWS_CanTp_00160
-	-	SWS_CanTp_00166
-	-	SWS_CanTp_00167
-	-	SWS_CanTp_00168
-	-	SWS_CanTp_00169
-	-	SWS_CanTp_00176
-	-	SWS_CanTp_00177
-	-	SWS_CanTp_00184
-	-	SWS_CanTp_00190
-	-	SWS_CanTp_00199
-	-	SWS_CanTp_00200
-	-	SWS_CanTp_00202
-	-	SWS_CanTp_00204
-	-	SWS_CanTp_00205

-	-	SWS_CanTp_00206
-	-	SWS_CanTp_00209
-	-	SWS_CanTp_00210
-	-	SWS_CanTp_00211
-	-	SWS_CanTp_00212
-	-	SWS_CanTp_00213
-	-	SWS_CanTp_00214
-	-	SWS_CanTp_00215
-	-	SWS_CanTp_00216
-	-	SWS_CanTp_00217
-	-	SWS_CanTp_00221
-	-	SWS_CanTp_00222
-	-	SWS_CanTp_00223
-	-	SWS_CanTp_00224
-	-	SWS_CanTp_00225
-	-	SWS_CanTp_00229
-	-	SWS_CanTp_00231
-	-	SWS_CanTp_00232
-	-	SWS_CanTp_00235
-	-	SWS_CanTp_00236
-	-	SWS_CanTp_00238
-	-	SWS_CanTp_00242
-	-	SWS_CanTp_00243
-	-	SWS_CanTp_00246
-	-	SWS_CanTp_00248
-	-	SWS_CanTp_00253
-	-	SWS_CanTp_00254
-	-	SWS_CanTp_00255
-	-	SWS_CanTp_00256
-	-	SWS_CanTp_00257
-	-	SWS_CanTp_00260
-	-	SWS_CanTp_00261
-	-	SWS_CanTp_00262
-	-	SWS_CanTp_00263
-	-	SWS_CanTp_00264
-	-	SWS_CanTp_00267
-	-	SWS_CanTp_00269
-	-	SWS_CanTp_00271
-	-	SWS_CanTp_00272

-	-	SWS_CanTp_00273
-	-	SWS_CanTp_00274
-	-	SWS_CanTp_00277
-	-	SWS_CanTp_00278
-	-	SWS_CanTp_00280
-	-	SWS_CanTp_00281
-	-	SWS_CanTp_00282
-	-	SWS_CanTp_00283
-	-	SWS_CanTp_00284
-	-	SWS_CanTp_00285
-	-	SWS_CanTp_00286
-	-	SWS_CanTp_00287
-	-	SWS_CanTp_00288
-	-	SWS_CanTp_00289
-	-	SWS_CanTp_00291
-	-	SWS_CanTp_00293
-	-	SWS_CanTp_00294
-	-	SWS_CanTp_00296
-	-	SWS_CanTp_00298
-	-	SWS_CanTp_00299
-	-	SWS_CanTp_00300
-	-	SWS_CanTp_00302
-	-	SWS_CanTp_00303
-	-	SWS_CanTp_00304
-	-	SWS_CanTp_00305
-	-	SWS_CanTp_00309
-	-	SWS_CanTp_00310
-	-	SWS_CanTp_00311
-	-	SWS_CanTp_00312
-	-	SWS_CanTp_00313
-	-	SWS_CanTp_00314
-	-	SWS_CanTp_00315
-	-	SWS_CanTp_00316
-	-	SWS_CanTp_00317
-	-	SWS_CanTp_00318
-	-	SWS_CanTp_00319
-	-	SWS_CanTp_00320
-	-	SWS_CanTp_00321
-	-	SWS_CanTp_00322

-	-	SWS_CanTp_00323
-	-	SWS_CanTp_00324
-	-	SWS_CanTp_00325
-	-	SWS_CanTp_00328
-	-	SWS_CanTp_00329
-	-	SWS_CanTp_00330
-	-	SWS_CanTp_00331
-	-	SWS_CanTp_00332
-	-	SWS_CanTp_00333
-	-	SWS_CanTp_00334
-	-	SWS_CanTp_00335
-	-	SWS_CanTp_00336
-	-	SWS_CanTp_00337
-	-	SWS_CanTp_00338
-	-	SWS_CanTp_00339
-	-	SWS_CanTp_00340
-	-	SWS_CanTp_00341
-	-	SWS_CanTp_00342
-	-	SWS_CanTp_00343
-	-	SWS_CanTp_00344
-	-	SWS_CanTp_00345
-	-	SWS_CanTp_00346
-	-	SWS_CanTp_00347
-	-	SWS_CanTp_00348
-	-	SWS_CanTp_00349
BSW00324	-	SWS_CanTp_00327
BSW00420	-	SWS_CanTp_00327
BSW00431	-	SWS_CanTp_00327
BSW00434	-	SWS_CanTp_00327
BSW381	-	SWS_CanTp_00001
BSW382	-	SWS_CanTp_00327
BSW383	-	SWS_CanTp_00327
BSW397	-	SWS_CanTp_00327
BSW398	-	SWS_CanTp_00327
BSW399	-	SWS_CanTp_00327
BSW400	-	SWS_CanTp_00327
BSW406	-	SWS_CanTp_00161
BSW412	-	SWS_CanTp_00156
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a	SWS_CanTp_00327



	defined configuration for all supported platforms.	
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_CanTp_00208
SRS_BSW_00158	All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation	SWS_CanTp_00001, SWS_CanTp_00156
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_CanTp_00146
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_CanTp_00327
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_CanTp_00327
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_CanTp_00147
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_CanTp_00327
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_CanTp_00327
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_CanTp_00327
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only import the necessary information	SWS_CanTp_00156
SRS_BSW_00307	Global variables naming convention	SWS_CanTp_00327
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_CanTp_00327
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_CanTp_00327
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_CanTp_00327
SRS_BSW_00326	-	SWS_CanTp_00327
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_CanTp_00327
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_CanTp_00327

SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_CanTp_00010
SRS_BSW_00339	Reporting of production relevant error status	SWS_CanTp_00008
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_CanTp_00327
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_CanTp_00327
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_CanTp_00327
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_CanTp_00001
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_CanTp_00156
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_CanTp_00327
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_CanTp_00002
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_CanTp_00208
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_CanTp_00327
SRS_BSW_00370	-	SWS_CanTp_00156
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_CanTp_00164
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_CanTp_00327
SRS_BSW_00376	-	SWS_CanTp_00164
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_CanTp_00327
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_CanTp_00327
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_CanTp_00327
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_CanTp_00327
SRS_BSW_00414	The init function may have parameters	SWS_CanTp_00208

SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_CanTp_00327
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_CanTp_00327
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_CanTp_00327
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	SWS_CanTp_00327
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_CanTp_00327
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_CanTp_00327
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_CanTp_00164
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_CanTp_00327
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_CanTp_00327
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_CanTp_00327
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_CanTp_00327
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_CanTp_00327
SRS_BSW_00435	-	SWS_CanTp_00156
SRS_BSW_00436	-	SWS_CanTp_00156
SRS_Can_01065	The AUTOSAR CAN Transport Layer shall be based on ISO 15765-2 and 15765-4 specifications	SWS_CanTp_00033
SRS_Can_01066	The AUTOSAR CAN Transport Layer shall be statically configurable to support either single or multiple connections in an optimizing way	SWS_CanTp_00096, SWS_CanTp_00120, SWS_CanTp_00121, SWS_CanTp_00122, SWS_CanTp_00123, SWS_CanTp_00124
SRS_Can_01068	The CAN Transport Layer shall identify each N-SDU with a unique identifier.	SWS_CanTp_00035
SRS_Can_01069	CAN address information and N-SDU identifier mapping	SWS_CanTp_00035

SRS_Can_01071	The CAN Transport Layer shall identify each N-PDU (also called L-SDU) with a unique identifier	SWS_CanTp_00035
SRS_Can_01073	The CAN Transport Layer shall be statically configured to pad unused bytes of PDU	SWS_CanTp_00116
SRS_Can_01075	The CAN Transport Layer shall implement an interface for initialization	SWS_CanTp_00030, SWS_CanTp_00170
SRS_Can_01076	The CAN Transport Layer services shall not be operational before initializing the module	SWS_CanTp_00031
SRS_Can_01078	The AUTOSAR CAN Transport Layer shall support the ISO 15765-2 addressing formats	SWS_CanTp_00035
SRS_Can_01082	Error handling	SWS_CanTp_00057
SRS_Can_01086	Data padding value of unused bytes	SWS_CanTp_00059
SRS_Can_01117	The CAN Transport Layer shall support half-duplex communication for TP channels	SWS_CanTp_00057, SWS_CanTp_00290
SRS_Can_01149	The CAN Transport Layer shall support full-duplex communication for TP channels	SWS_CanTp_00057, SWS_CanTp_00290

## 7 Functional specification

This section provides a description of the CAN Transport Layer functionality. It explains the services provided to the upper and lower layers and the internal behavior of the CAN Transport Layer.

The CanTp module offers services for segmentation, transmission with flow control, and reassembly of messages. Its main purpose is to transmit and receive messages that may or may not fit into a single CAN frame. Messages that do not fit into a single CAN frame are segmented into multiple parts, such that each can be transmitted in a single CAN frame.

While reading this document, it is necessary to bear in mind, that this module will follow the recommendations ISO 15765-2 (OEM enhanced diagnostics [14]) and should be able to fulfill ISO 15765-4 (Requirements for emissions-related systems [16]).

**[SWS\_CanTp\_00033]** [If a recommendation of ISO 15765-2 is not explicitly excluded in the SWS, the CanTp module shall follow this recommendation. ] (SRS\_Can\_01065)

For further descriptions of SF, FF, CF and FC frames, network layer timing parameters, and further functionalities of CAN Transport Layer please refer to the ISO 15765-2 specification [14].

ISO 15765-4 is a particular case of ISO-15765-2. Therefore, the CAN Transport Layer will be configurable in order to be able to adapt the module to all ISO 15765-4 use cases (e.g. specific timing, padding configuration, addressing mode). See chapter 10, Configuration specification, for details.

### 7.1 Services provided to upper layer

The service interface of the CanTp module can be divided into the following main categories:

- Initialization and shutdown
- Communication services

The following paragraphs describe the functionality of each services category.

#### 7.1.1 Initialization and shutdown

**[SWS\_CanTp\_00027]** [The CanTp module shall have two internal states, CANTP\_OFF and CANTP\_ON. ] ( )

**[SWS\_CanTp\_00253]** [The `CANTP_OFF` and `CANTP_ON` states shall be available for debugging (chapter 7.7 details implementation of debugging concept). ] ( )

**[SWS\_CanTp\_00168]** [The CanTp module shall be in the `CANTP_OFF` state after power up. ] ( )

**[SWS\_CanTp\_00169]** [In the state `CANTP_OFF`, the CanTp shall allow an update of the postbuild configuration. ] ( )

**[SWS\_CanTp\_00170]** [The CanTp module shall change to the internal state `CANTP_ON` when the CanTp has been successfully initialized with `CanTp_Init()`.] (SRS\_Can\_01075)

**[SWS\_CanTp\_00238]** [The CanTp module shall perform segmentation and reassembly tasks only when the CanTp is in the `CANTP_ON` state. ] ( )

**[SWS\_CanTp\_00030]** [The function `CanTp_Init` shall initialize all global variables of the module and sets all transport protocol connections in a sub-state of `CANTP_ON`, in which neither segmented transmission nor segmented reception are in progress (Rx thread in state `CANTP_RX_WAIT` and Tx thread in state `CANTP_TX_WAIT`). ] (SRS\_Can\_01075)

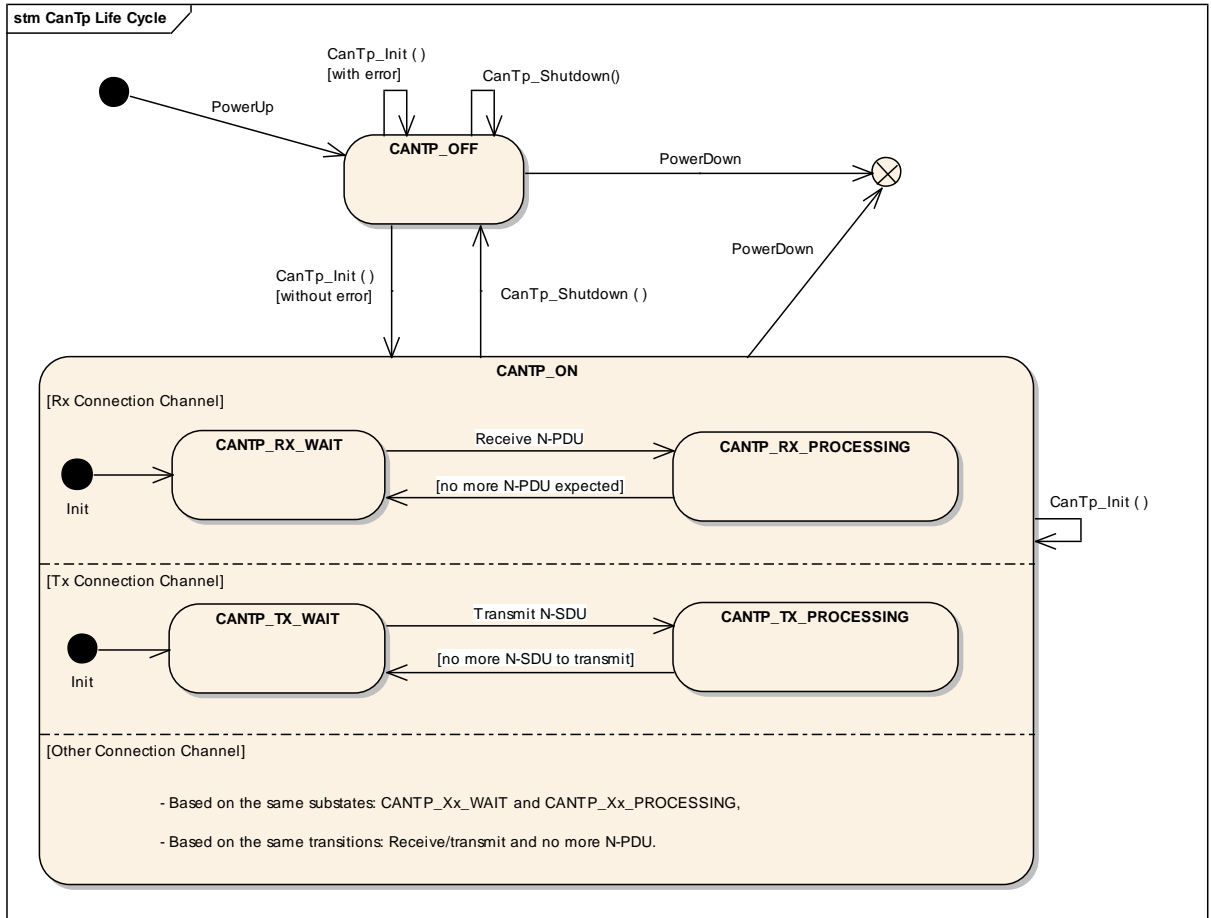
**[SWS\_CanTp\_00031]** [If development error detection for the CanTp module is enabled the CanTp module shall raise an error (`CANTP_E_UNINIT`) when the PDU Router or CAN Interface tries to use any function (except `CanTp_GetVersionInfo`) before the function `CanTp_Init` has been called. ] (SRS\_Can\_01076)

**[SWS\_CanTp\_00111]** [If called when the CanTp module is in the global state `CANTP_ON`, the function `CanTp_Init` shall return the module to state Idle (state = `CANTP_ON`, but neither transmission nor reception are in progress). ] ( )

**[SWS\_CanTp\_00273]** [The CanTp module shall loose all current connections if `CanTp_Init` is called when CanTp module is in the global state `CANTP_ON`. ] ( )

**[SWS\_CanTp\_00010]** [The function `CanTp_Shutdown` shall stop the CanTp module properly. ] (SRS\_BSW\_00336)

The following figure summarizes all of the above requirements:



**Figure 4: CAN Transport Layer life cycle**

**7.1.2 Transmit request**

The transmit operation, CanTp\_Transmit(), will allow upper layers to ask for data transfer using CAN transport protocol facilities (segmentation, extended addressing format and so on).

**[SWS\_CanTp\_00176]** [The function CanTp\_Transmit() shall be asynchronous. ] ( )

**[SWS\_CanTp\_00177]**[After the transmit request was accepted, the CanTp module shall notify its upper layer if the N-SDU transfer is fully processed (successfully or not). ] ( )

**7.1.3 Transmit cancellation**

The transmit cancellation feature allows the upper layer to cancel a transmission in progress.

**Use case:** Cancel a diagnostic transmission due to the reception of another diagnostic protocol with higher priority.

**[SWS\_CanTp\_00242]** [This feature shall be (de)activated by static configuration (parameter CanTpTc). ] ( )

**[SWS\_CanTp\_00274]** [Transmit Cancellation is triggered by the call of CanTp\_CancelTransmit().] ( )

**[SWS\_CanTp\_00243]** [After the call of the service CanTp\_CancelTransmit(), the transfer on this connection shall be aborted. ] ( )

**Note:** The Api PduR\_CanTpTxConfirmation() shall be called after a transmit cancellation with value E\_NOT\_OK.(see also [SWS\\_CanTp\\_00255](#))

Note that if a transfer is in progress, that will generate a time-out error on the receiver side.

## 7.2 Services provided to the lower layer

According to the AUTOSAR specification of the communication stack, the CAN Transport Layer provides the following two callback functions to the Can interface: CanTp\_TxConfirmation() and CanTp\_RxIndication().

### 7.2.1 Transmit confirmation

The CanIf module shall call the transmit confirmation function to notify the CAN Transport Layer that a CAN frame transmission, requested by the CanTp, has been performed successfully. The L-PDU identifier is associated with the call in order to identify the corresponding transmission.

**[SWS\_CanTp\_00075]** [If the transmit confirmation is not received after a maximum time (equal to N\_A<sub>s</sub>), the CanTp module shall act as if it had received an unsuccessful transmission confirmation and any late confirmation shall be ignored. The CanTp module shall cancel (internally) the failed transmission. ] ( )

**[SWS\_CanTp\_00076]** [For confirmation calls, the CanTp module shall provide the function CanTp\_TxConfirmation().] ( )

### 7.2.2 Reception indication

The CanIf module shall call the reception indication function to notify the CanTp module that a new CAN N-PDU frame (i.e. a transport protocol frame) has been received.

The reception indication can be performed in ISR context according to CanIf configuration.



**[SWS\_CanTp\_00078]** [For reception indication, the CanTp module shall provide `CanTp_RxIndication().`] ( )

### 7.3 Internal behavior

The internal operation of the CAN Transport Layer provides basic mechanisms in order to perform the main purpose of this module, which is to transfer messages in a single CAN frame or in multiple CAN frames.

The entire behavior of the CAN Transport Layer will be event triggered, so that CanTp can process transfer of N-SDU (respectively L-SDU) coming from the PDU Router (respectively CAN Interface) directly.

#### 7.3.1 N-SDU Reception

**[SWS\_CanTp\_00079]** [When receiving an SF or an FF N-PDU, the CanTp module shall notify the upper layer (PDU Router) about this reception using the `PduR_CanTpStartOfReception` function. ] ( )

Note: The upper layer will reserve and lock a buffer for reception of the N-SDU.

**[SWS\_CanTp\_00329]** [CanTp shall provide the content of the FF/SF to PduR using the parameter `TpSduInfoPtr` of `PduR_CanTpStartOfReception().` ] ( )

**[SWS\_CanTp\_00330]** [ When `CanTp_RxIndication` is called for a SF or FF N-PDU with `MetaData` (indicating a generic connection), the CanTp module shall store the addressing information contained in the `MetaData` of the PDU (at the end of the data referenced by `PduInfoPtr`) and use this information for the initiation of the connection to the upper layer, for transmission of FC N-PDUs and for identification of CF N-PDUs. The addressing information in the `MetaData` depends on the addressing format:

- Normal, Extended, Mixed 11 bit: OEM dependent
- Normal fixed, Mixed 29 bit: `N_SA`, `N_TA`] ( )

**[SWS\_CanTp\_00331]** [ When calling `PduR_CanTpStartOfReception()` for a generic connection (N-SDU with `MetaData`), the CanTp module shall forward the extracted addressing information via the `MetaData` of the N-SDU (at the end of the data referenced by `info`). The addressing information in the `MetaData` depends on the addressing format:

- Normal: none
- Extended: `N_TA`
- Mixed 11 bit: `N_AE`
- Normal fixed: `N_SA`, `N_TA`

- Mixed 29 bit: N\_SA, N\_TA, N\_AE ] ( )

**[SWS\_CanTp\_00332]** [ When calling `CanIf_Transmit()` for an FC on a generic connection (N-PDU with MetaData), the CanTp module shall provide the stored addressing information via the MetaData of the N-PDU (at the end of the data referenced by `PduInfoPtr`). The addressing information in the MetaData depends on the addressing format:

- Normal, Extended, Mixed 11 bit: OEM dependent
- Normal fixed, Mixed 29 bit: N\_SA (saved N\_TA), N\_TA (saved N\_SA) ] ( )

**[SWS\_CanTp\_00333]** [ When `CanTp_RxIndication` is called for a CF on a generic connection (N-PDU with MetaData), the CanTp module shall check the addressing information contained in the MetaData of the N-PDU (at the end of the data referenced by `PduInfoPtr`) against the stored values from the FF. ] ( )

**[SWS\_CanTp\_00166]** [At the reception of a FF or last CF of a block, the CanTp module shall start a time-out `N_Br` before calling `PduR_CanTpStartOfReception` or `PduR_CanTpCopyRxData`. ] ( )

**[SWS\_CanTp\_00080]** [The available Rx buffer size is reported to the CanTp in the output pointer parameter of the `PduR_CanTpStartOfReception()` service. The available Rx buffer can be smaller than the expected N-SDU data length. ] ( )

Note: If the upper layer cannot make a buffer available because of an error (e.g. in the gateway case it may indicate that the transport session to the destination network has been broken) or a resource limitation (e.g. N-SDU length exceeds the maximum buffer size of the upper layer), the `PduR_CanTpStartOfReception()` function returns `BUFREQ_E_NOT_OK` or `BUFREQ_E_OVFL`.

**[SWS\_CanTp\_00081]** [After the reception of a First Frame or Single Frame, if the function `PduR_CanTpStartOfReception()` returns `BUFREQ_E_NOT_OK` to the CanTp module, the CanTp module shall abort the reception of this N-SDU. No Flow Control will be sent and `PduR_CanTpRxIndication()` will not be called in this case. ] ( )

**[SWS\_CanTp\_00318]** [After the reception of a First Frame or Single Frame, if the function `PduR_CanTpStartOfReception()` returns `BUFREQ_E_OVFL` to the CanTp module, the CanTp module shall send a Flow Control N-PDU with overflow status (FC(OVFLW)) and abort the N-SDU reception. ] ( )

**[SWS\_CanTp\_00339]** [After the reception of a First Frame or Single Frame, if the function `PduR_CanTpStartOfReception()` returns `BUFREQ_OK` with a smaller available buffer size than needed for the already received data, the CanTp module

shall abort the reception of the N-SDU and call `PduR_CanTpRxIndication()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00082]** [After the reception of a First Frame, if the function `PduR_CanTpStartOfReception()` returns `BUFREQ_OK` with a smaller available buffer size than needed for the next block, the CanTp module shall start the timer `N_Br`.] ( )

**[SWS\_CanTp\_00325]** [If the function `PduR_CanTpCopyRxData()` called after reception of the last Consecutive Frame of a block returns `BUFREQ_OK`, but the remaining buffer is not sufficient for the reception of the next block, the CanTp module shall start the timer `N_Br`.] ( )

**[SWS\_CanTp\_00222]** [ While the timer `N_Br` is active, the CanTp module shall call the service `PduR_CanTpCopyRxData()` with a data length 0 (zero) and `NULL_PTR` as data buffer during each processing of the `MainFunction`. ] ( )

Note: ISO 15765-2 specification defines the following performance requirement:  
 $(N\_Br + N\_Ar) < 0.9 * N\_Bs$  timeout.

**[SWS\_CanTp\_00341]** [If the `N_Br` timer expires and the available buffer size is still not big enough, the CanTp module shall send a new FC(WAIT) to suspend the N-SDU reception and reload the `N_Br` timer.] ( )

**[SWS\_CanTp\_00223]** [The CanTp module shall send a maximum of `WFTmax` consecutive FC(WAIT) N-PDU. If this number is reached, the CanTp module shall abort the reception of this N-SDU (the receiver did not send any FC N-PDU, so the `N_Bs` timer expires on the sender side and then the transmission is aborted) and a receiving indication with `E_NOT_OK` occurs. ] ( )

**[SWS\_CanTp\_00311]** [In case of `N_Ar` timeout occurrence (no confirmation from CAN driver for any of the FC frame sent) the CanTp module shall abort reception and notify the upper layer of this failure by calling the indication function `PduR_CanTpRxIndication()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00224]** [When the Rx buffer is large enough for the next block (directly after the First Frame or the last Consecutive Frame of a block, or after repeated calls to `PduR_CanTpCopyRxData()` according to [SWS\\_CanTp\\_00222](#)), the CanTp module shall send a Flow Control N-PDU with `ClearToSend` status (FC(CTS)) and shall then expect the reception of Consecutive Frame N-PDUs.] ( )

**[SWS\_CanTp\_00269]** [After reception of each Consecutive Frame the CanTp shall call the `PduR_CanTpCopyRxData()` function with a `PduInfo` pointer containing data buffer and 6 or 7 data length (or less in case of the last CF). The output pointer

parameter provides CanTp with available Rx buffer size after data have been copied.  
] ( )

**[SWS\_CanTp\_00312]** [The CanTp module shall start a time-out N\_Cr at each indication of CF reception (except the last one in a block) and at each confirmation of a FC transmission that initiate a CF transmission on the sender side (FC with FS=CTS). ] ( )

**[SWS\_CanTp\_00313]** [In case of N\_Cr timeout occurrence the CanTp module shall abort reception and notify the upper layer of this failure by calling the indication function `PduR_CanTpRxIndication()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00271]** [If the `PduR_CanTpCopyRxData()` returns `BUFREQ_E_NOT_OK` after reception of a Consecutive Frame in a block the CanTp shall abort the reception of N-SDU and notify the PduR module by calling the `PduR_CanTpRxIndication()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00314]** [The CanTp shall check the correctness of each SN received during a segmented reception. In case of wrong SN received the CanTp module shall abort reception and notify the upper layer of this failure by calling the indication function `PduR_CanTpRxIndication()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00084]** [When the transport reception session is completed (successfully or not) the CanTp module shall call the upper layer notification service `PduR_CanTpRxIndication()`.] ( )

**[SWS\_CanTp\_00277]** [With regard to FF N-PDU reception, the content of the Flow Control N-PDU depends on the `PduR_CanTpStartOfReception()` service result.  
] ( )

**[SWS\_CanTp\_00064]** [Furthermore, it should be noted that when receiving a FF N-PDU, the Flow Control shall only be sent after having the result of the `PduR_CanTpStartOfReception()` service. ] ( )

**[SWS\_CanTp\_00278]** [It is important to note that FC N-PDU will only be sent after every block, composed of a number BS (Block Size) of consecutive frames. ] ( )

**[SWS\_CanTp\_00067]** [The CanTp shall use the same value for the BS and STmin parameters on each FC sent during a segmented reception. Different values of these parameters can be used on different N-SDUs reception. ] ( )

**[SWS\_CanTp\_00342]** [CanTp shall terminate the current reception connection when `CanIf_Transmit()` returns `E_NOT_OK` when transmitting an FC.] ( )

### 7.3.2 N-SDU Transmission

As described in chapter 0, the upper layer asks for the transmission of a N-SDU by calling `CanTp_Transmit()`. The parameters of `CanTp_Transmit()` describe the CAN NSdulid and the full Tx N-SDU length to be sent .

**[SWS\_CanTp\_00225]** [For specific connections that do not use MetaData, the function `CanTp_Transmit` shall only use the full `SduLength` information and shall not use the available N-SDU data buffer in order to prepare Single Frame or First Frame PCI. ] ( )

**[SWS\_CanTp\_00334]** [ When `CanTp_Transmit` is called for an N-SDU with MetaData, the `CanTp` module shall store the addressing information contained in the MetaData of the N-SDU (at the end of the data referenced by `CanTpTxInfoPtr`) and use this information for transmission of SF, FF, and CF N-PDUs and for identification of FC N-PDUs. The addressing information in the MedataData depends on the addressing format:

- Normal: none
- Extended: N\_TA
- Mixed 11 bit: N\_AE
- Normal fixed: N\_SA, N\_TA
- Mixed 29 bit: N\_SA, N\_TA, N\_AE. ] ( )

**[SWS\_CanTp\_00335]** [ When calling `CanIf_Transmit()` for an SF, FF, or CF of a generic connection (N-PDU with MetaData), the `CanTp` module shall provide the stored addressing information via MetaData of the N-PDU (at the end of the data referenced by `PduInfoPtr`). The addressing information in the MetaData depends on the addressing format:

- Normal, Extended, Mixed 11 bit: OEM dependent
- Normal fixed, Mixed 29 bit: N\_SA, N\_TA. ] ( )

**[SWS\_CanTp\_00336]** [ When `CanTp_RxIndication` is called for an FC on a generic connection (N-PDU with MetaData), the `CanTp` module shall check the addressing information contained in the MetaData (at the end of the data referenced by `PduInfoPtr`) against the stored values. ] ( )

**[SWS\_CanTp\_00167]** [After a transmission request from upper layer, the `CanTp` module shall start time-out `N-Cs` before the call of `PduR_CanTpCopyTxData`. If no data is available before the timer elapsed, the `CanTp` module shall abort the communication. ] ( )

**[SWS\_CanTp\_00086]** [The `CanTp` module shall call the `PduR_CanTpCopyTxData` service for each segment that is sent (SF, FF and CF). The upper layer copy the transmit data on the `PduInfoType` structure (address of a data buffer and up to 6 or 7 bytes as data length depending on frame type). ] ( )

**[SWS\_CanTp\_00272]** [The API `PduR_CanTpCopyTxData()` contains a parameter used for the recovery mechanism – ‘retry’. Because ISO 15765-2 does not support such a mechanism, the CAN Transport Layer does not implement any kind of recovery. Thus, the parameter is always set to NULL pointer. ] ( )

If the upper layer cannot make Tx data available because of an error (e.g. in gateway case it may indicate that the transport session from the source network was terminated), the `PduR_CanTpCopyTxData()` function returns `BUFREQ_E_NOT_OK`.

**[SWS\_CanTp\_00087]** [If `PduR_CanTpCopyTxData()` returns `BUFREQ_E_NOT_OK`, the CanTp module shall abort the transmit request and notify the upper layer of this failure by calling the callback function `PduR_CanTpTxConfirmation()` with the result `E_NOT_OK`. ] ( )

**Note:** If upper layer temporarily has no Tx buffer available, the `PduR_CanTpCopyTxData()` function returns `BUFREQ_E_BUSY`.

**[SWS\_CanTp\_00184]** [If the `PduR_CanTpCopyTxData()` function returns `BUFREQ_E_BUSY`, the CanTp module shall later (implementation specific) retry to copy the data. ] ( )

Note: If no data is available before the expiration of the `N_Cs` timer (ISO 15765-2 specification defines the following performance requirement:  $(N_Cs + N_As) < 0.9 * N_Cr$  timeout), the CanTp module shall abort this transmission session.

**[SWS\_CanTp\_00280]** [If data is not available within `N_Cs` timeout the CanTp module shall notify the upper layer of this failure by calling the callback function `PduR_CanTpTxConfirmation` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00089]** [When Tx data is available, the CanTp module shall resume the transmission of the N-SDU. ] ( )

**[SWS\_CanTp\_00310]** [In case of `N_As` timeout occurrence (no confirmation from CAN driver) the CanTp module shall notify the upper layer by calling the callback function `PduR_CanTpTxConfirmation()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00309]** [If a FC frame is received with the FS set to `OVFLW` the CanTp module shall abort the transmit request and notify the upper layer by calling the callback function `PduR_CanTpTxConfirmation()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00317]** [If a FC frame is received with an invalid FS the CanTp module shall abort the transmission of this message and notify the upper layer by

calling the callback function `PduR_CanTpTxConfirmation()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00315]** [The CanTp module shall start a timeout observation for `N_Bs` time at confirmation of the FF transmission, last CF of a block transmission and at each indication of FC with `FS=WT` (i.e. time until reception of the next FC). ] ( )

**[SWS\_CanTp\_00316]** [In case of `N_Bs` timeout occurrence the CanTp module shall abort transmission of this message and notify the upper layer by calling the callback function `PduR_CanTpTxConfirmation()` with the result `E_NOT_OK`. ] ( )

**[SWS\_CanTp\_00090]** [When the transport transmission session is successfully completed, the CanTp module shall call a notification service of the upper layer, `PduR_CanTpTxConfirmation()`, with the result `E_OK`. ] ( )

**[SWS\_CanTp\_00343]** [CanTp shall terminate the current transmission connection when `CanIf_Transmit()` returns `E_NOT_OK` when transmitting an SF, FF, or CF. ] ( )

### 7.3.3 Buffer strategy

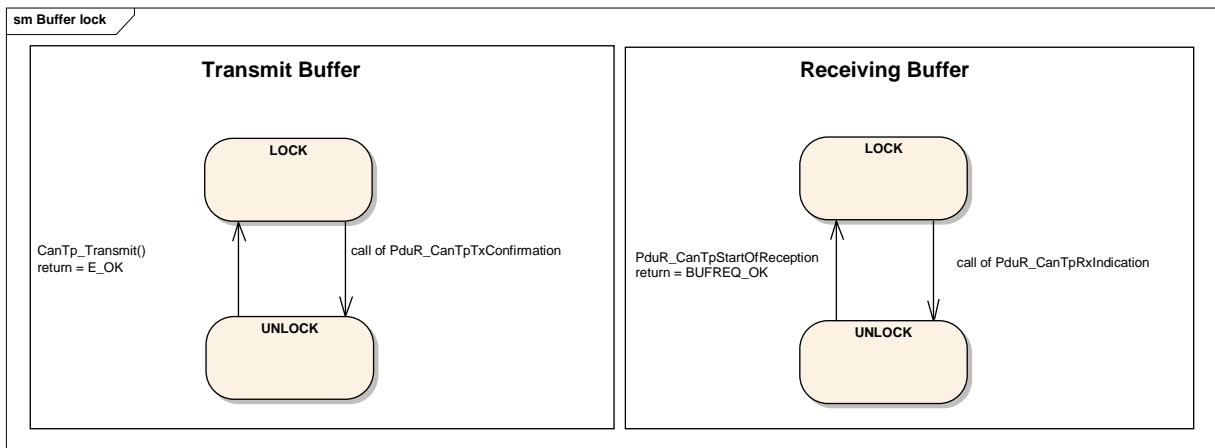
Because CanTp has no buffering capability, the N-SDU payload, which is to be transmitted, is not copied internally and the N-PDU received is not reassembled internally.

The CAN Transport Layer works directly on the memory area of the upper layers (e.g. PduR, DCM, or COM). To access these memory areas, the CAN Transport Layer uses the `PduR_CanTpCopyTxData()` or `PduR_CanTpCopyRxData()` functions.

Thus, to guarantee data consistency, the upper layer should lock this memory area until an indication occurs.

When a transmit buffer is locked, the upper layer must not write data inside the buffer area.

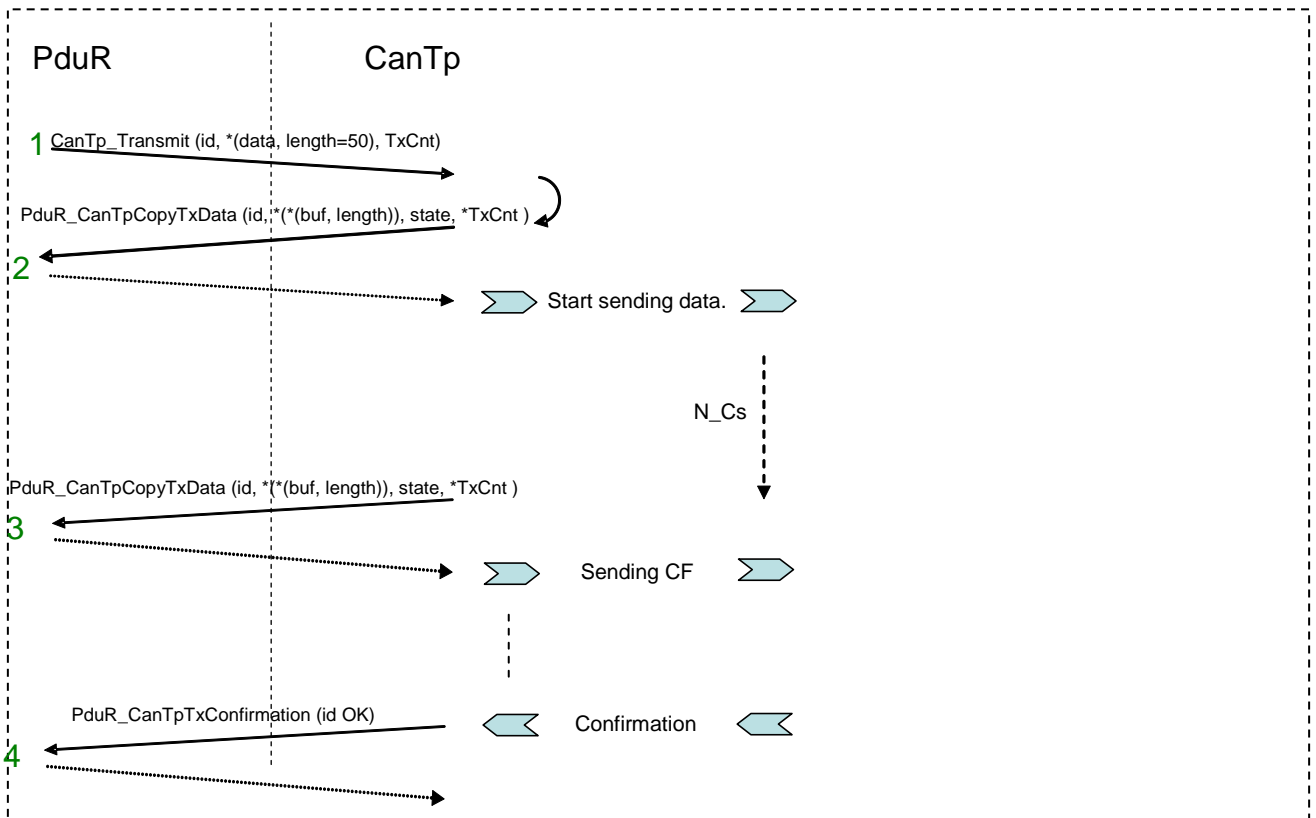
When a receiving buffer is locked the CAN Transport Layer does not guarantee data consistency of the buffer. The upper layer should neither read nor write data in the buffer area.



**Figure 5: Tx and Rx Buffer locking**

It is assumed that upper layer module has locked the buffer when it returns a status BUFREQ\_OK to a PduR\_CanTpStartOfReception() call or when CanTp\_Transmit() returns E\_OK and shall keep the buffer locked until a confirmation or indication (PduR\_CanTpTxConfirmation() or PduR\_CanTpRxIndication() call) occurs.

The following figure provides an example, to summarize the process of sending a frame, with a length of 50 bytes.

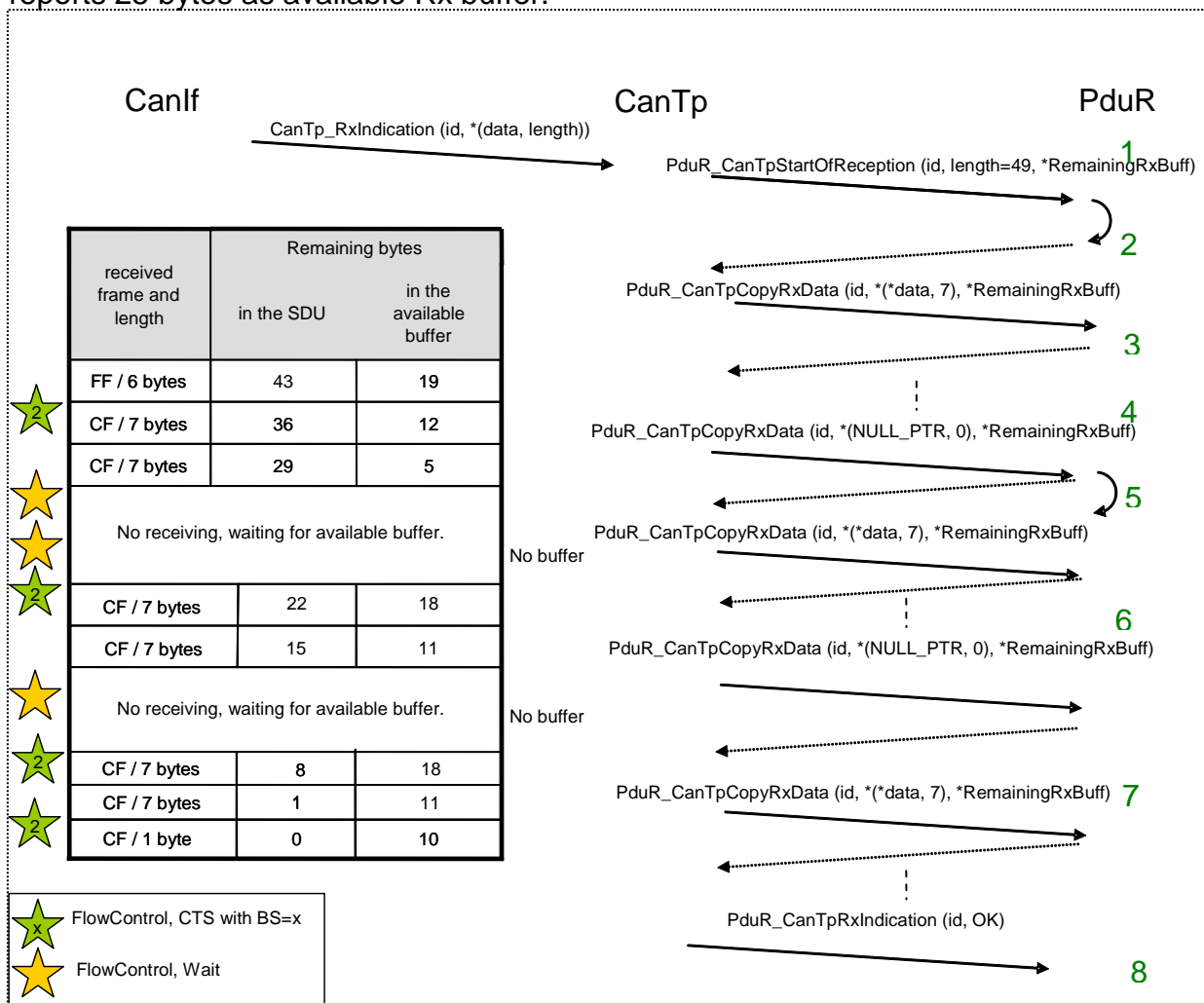


**Figure 6: Example of transmit process**



- 1: The PduR asks for the transmission of 50 data bytes;
- 2: The CanTp asks the PduR for the payload data; the CanTp send the First Frame;
- 3: The CanTp send the rest of payload data as sequences of Consecutive Frames; 6 or 7 payload data bytes are copied by the upper layer on each CF;
- 4: The CanTp confirms transmission of the payload data.

The next figure is an example of an N-SDU receiving 49 bytes; the upper layer reports 25 bytes as available Rx buffer.



**Figure 7: Example of receiving process**

- 1: The CanIf notifies a new reception with `CanTp_RxIndication()`.The CanTp forwards this notification to the PduR;
- 2: The PduR returns an available buffer size of 25 bytes, CanTp sends a FlowControl CTS to the originator;
- 3: The CanTp provides the data of each received frame to the PduR and monitors the remaining buffer size. After the second consecutive frame, the remaining buffer size is not enough for the next block (two Consecutive Frames);
- 4: The CanTp asks the PduR for the remaining buffer size by calling `PduR_CanTpCopyRxData()` with 0 as data length and `NULL_PTR` as data, and

sends a FlowControl Wait to the originator. This is done until sufficient buffer for the next block is available;

**5:** When the buffer size is finally sufficient for the next block, the CanTp will send a FlowControl CTS to the originator and continues the reception of the next Consecutive Frames block;

**6:** After copying the last consecutive frame of the block, the remaining buffer is too low for the next block, so CanTp again sends wait frames and monitors the remaining buffer size;

**7:** When the buffer for the last block is available, CanTp will continue the reception;

**8:** The CanTp informs the PduR of the end of reception by a call to PduR\_CanTpRxIndication().

The CAN Transport Layer will compute the BS values (See [SWS\\_CanTp\\_00067](#)) depending on:

- maximum configured value for this N-SDU,
- number of free bytes inside the available buffer,
- amount of receiving bytes.

If the BS value is equal to 0 the buffer should be sized to a value equal or larger than the number of bytes to be received.

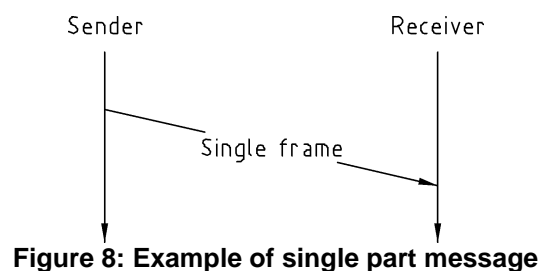
### 7.3.4 Protocol parameter setting services

**[SWS\_CanTp\_00091]** [The CanTp module shall support optional primitives (proposed in ISO 15765-2 specification) for the dynamic setting of some transport protocol internal parameters (STmin and BS) by application.

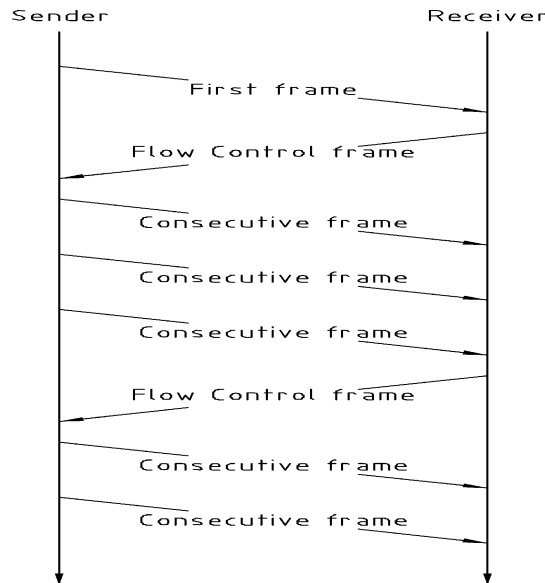
The BS value is only a maximum value. For reasons of buffer length, the CAN Transport Layer can adapt the BS value within the limit of the configured maximum value. ] ( )

### 7.3.5 Tx and Rx data flow

The following figures show examples of an un-segmented message transmission and a segmented one.



**Figure 8: Example of single part message**



**Figure 9: Example of multiple parts message**

Flow control is used to adjust the sender to the capabilities of the receiver. The main usage of this transport protocol is peer-to-peer communication (i.e. 1 to 1 communication – physical addressing [14]).

**[SWS\_CanTp\_00092]** [The CanTp module shall provide 1 to n communication (i.e. functional addressing [14]), in the form of functionality to SF N-PDUs (and only SF N-SDU). ] ( )

The configuration tool shall check whether it is only SF N-PDUs that have been configured with a functional addressing property.

**[SWS\_CanTp\_00093]** [If a multiple segmented session occurs (on both receiver and sender side) with a handle whose communication type is functional, the CanTp module shall reject the request and generate, if the development error detection is enabled, a development error CANTP\_E\_INVALID\_TATYPE. ] ( )

### 7.3.6 Relationship between CAN NSduld and CAN LSduld

This chapter describes the connection that exists between CAN NSduld and CAN LSduld, in order to make transmission and reception of transport protocol data units possible.

**[SWS\_CanTp\_00035]** [A CAN NSduld shall only be linked to one CAN LSduld that is used to transmit SF, FF, FC and CF frames. ] (SRS\_Can\_01068, SRS\_Can\_01069, SRS\_Can\_01071, SRS\_Can\_01078)

**[SWS\_CanTp\_00281]** [However, if the message is configured to use an extended or a mixed addressing format, the CanTp module must fill the first byte of each transmitted segment (SF, FF and CF) with the N\_TA (in case of extended

addressing) or N\_AE (in case of mixed addressing) value. Therefore a CAN NSduld may also be related to a N\_TA or N\_AE value. ] ( )

**[SWS\_CanTp\_00282]** [FC protocol data units give receivers the possibility of controlling senders' data flow by authorizing or delaying transmission of subsequent CF N-PDUs. ] ( )

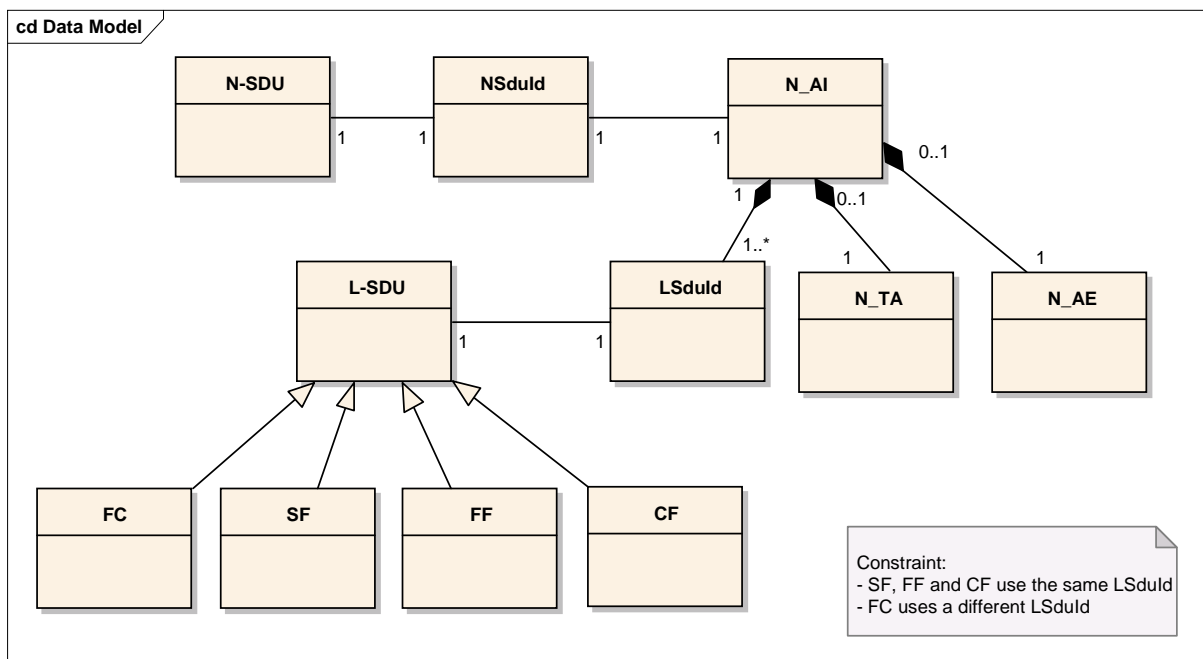
**[SWS\_CanTp\_00283]** [For extended addressing format, the first data byte of the FC also contains the N\_TA value or a unique combination of N\_TA and N\_TAtype value. For mixed addressing format, the first data byte of the FC contains the N\_AE value. ] ( )

**[SWS\_CanTp\_00094]** [Thus the CAN LSduld of a FC frame combined with its N\_TA value (e.g. the N\_AI) or with N\_AE value shall only identify one CAN NSduld. ] ( )

**[SWS\_CanTp\_00284]** [In the reception direction, the first data byte value of each (SF, FF or CF) transport protocol data unit will be used to determine the relevant N-SDU. ] ( )

**[SWS\_CanTp\_00095]** [Therefore, in extended addressing N-PDU reception, the CanTp module shall extract the N\_TA value to establish the related N-SDU. The same process shall be applied for mixed addressing mode in relation with N\_AE value. ] ( )

The following figure summarizes these discussions.



**Figure 10: Possible links between NSduld and LSduld**

### 7.3.7 Concurrent connection

The CAN Transport Layer is able to manage several connections simultaneously (e.g. a UDS and an OBD request can be received at the same time).

**[SWS\_CanTp\_00096]** [The CanTp module shall support several connections simultaneously. ] (SRS\_Can\_01066)

**[SWS\_CanTp\_00120]** [It shall be possible to configure concurrent connections in the CanTp module. ] (SRS\_Can\_01066)

**[SWS\_CanTp\_00285]** [The connection channels are only destined for CAN TP internal use, so they are not accessible externally. ] ( )

**[SWS\_CanTp\_00286]** [All the necessary information (Channel number, Timing parameter ...) is configured inside the CAN Transport Layer module. ] ( )

**[SWS\_CanTp\_00121]** [Each N-SDU is statically linked to one connection channel. This connection channel represents an internal path, for the transmission or receiving of the N-SDU. A connection channel is attached to one or more N-SDU. ] (SRS\_Can\_01066)

**[SWS\_CanTp\_00122]** [Each connection channel is independent of the other connection channels. This means that a connection channel uses its own resources, such as internal buffer, timer, or state machine. ] (SRS\_Can\_01066)

**[SWS\_CanTp\_00190]** [The CanTp module shall route the N-SDU through the correctly configured connection channel. ] ( )

**[SWS\_CanTp\_00287]** [ The CanTp module shall not accept the receiving or the transmission of N-SDU with the same identifier in parallel, because otherwise the received frames cannot be assigned to the correct connection. When only specific connections (without MetaData) are used, this requirement is enforced by the configuration, because each N-SDU is linked to only one connection channel.] ( )

If a user wants to dedicate a specific connection channel to only one N-SDU, they should assign this connection channel to one N-SDU only during the configuration process.

**[SWS\_CanTp\_00288]** [If a connection channel is assigned to multiple N-SDUs, then resources are shared between different N-SDUs, and the CAN Transport Layer will reject transmission or abort receiving, if no free connection channels are available. ] ( )

**[SWS\_CanTp\_00289]** [The number of connection channels is not directly configurable. It will be determined by the configuration tools during the configuration process, by analyzing the N-SDU/Channel routing table. ] ( )

**[SWS\_CanTp\_00123]** [If the configured transmit connection channel is in use (state `CANTP_TX_PROCESSING`), the CanTp module shall reject new transmission requests linked to this channel. To reject a transmission, CanTp returns `E_NOT_OK` when the upper layer asks for a transmission with the `CanTp_Transmit()` function.] (SRS\_Can\_01066)

**[SWS\_CanTp\_00124]** [When an SF or FF N-PDU without MetaData is received, and the corresponding connection channel is currently receiving the same connection (state `CANTP_RX_PROCESSING`, same `N_AI`), the CanTp module shall abort the reception in progress and shall process the received frame as the start of a new reception.

When an SF or FF N-PDU without MetaData is received for another connection (different `N_AI`) on an active connection channel, the SF or FF shall be ignored.] (SRS\_Can\_01066)

**[SWS\_CanTp\_00337]** [When an SF or FF N-PDU with MetaData (indicating a generic connection) is received, and the corresponding connection channel is currently receiving, the SF or FF shall be ignored. ] ( )

**[SWS\_CanTp\_00248]** [When a Tx N-PDU is used by two or more different connections on different channels, access to this N-PDU shall be serialized by using the TxConfirmation. An Rx N-PDU can only be used on two or more different connection channels if extended or mixed addressing is used in relation with this N-PDU or when it has MetaData (and thus belongs to a generic connection).] ( )

### 7.3.8 N-PDU padding

To guarantee complete compatibility with all upper layer requirements concerning the frame data length (e.g. OBD requires data length to always be set to 8 bytes, however UDS does not), the padding activation is configurable at pre-compile time per N-SDU by using either `CanTpRxPaddingActivation` for a Rx N-SDU or `CanTpTxPaddingActivation` for a Tx N-SDU.

**[SWS\_CanTp\_00116]** [In both padding and no padding modes, the CanTp module shall only transfer used data bytes to the upper layer. ] (SRS\_Can\_01073)

**[SWS\_CanTp\_00059]** [The value used for padding bytes is configurable via configuration parameter `CANTP_PADDING_BYTE` (see `ECUC_CanTp_00298`). ] (SRS\_Can\_01086)

**[SWS\_CanTp\_00344]** [If `CanTpRxPaddingActivation` is equal to `CANTP_ON` for an Rx N-SDU, the CanTp module shall only accept SF Rx N-PDUs or last CF Rx N-PDUs, belonging to that N-SDU, with a length of eight bytes (i.e. `PduInfoPtr.SduLength = 8`). ] ( )

**[SWS\_CanTp\_00345]** [If `CanTpRxPaddingActivation` is equal to `CANTP_ON` for an Rx N-SDU, and the CanTp module receives by means of `CanTp_RxIndication()` call an SF Rx N-PDU belonging to that N-SDU, with a length smaller than eight bytes (i.e. `PduInfoPtr.SduLength < 8`), the CanTp shall reject the reception. If the development error detection is enabled, a development error `CANTP_E_PADDING` shall be triggered. ] ( )

**[SWS\_CanTp\_00346]** [If `CanTpRxPaddingActivation` is equal to `CANTP_ON` for an Rx N-SDU, and the CanTp module receives by means of `CanTp_RxIndication()` call a last CF Rx N-PDU belonging to that N-SDU, with a length smaller than eight bytes (i.e. `PduInfoPtr.SduLength != 8`), the CanTp shall abort the ongoing reception by calling `PduR_CanTpRxIndication()` with the result `E_NOT_OK`. If the development error detection is enabled, a development error `CANTP_E_PADDING` shall be triggered. ] ( )

**[SWS\_CanTp\_00347]** [If `CanTpRxPaddingActivation` is equal to `CANTP_ON` for an Rx N-SDU, the CanTp module shall transmit FC N-PDUs with a length of eight bytes. Unused bytes in N-PDU shall be updated with `CANTP_PADDING_BYTE` (see [ECUC\\_CanTp\\_00298](#)). ] ( )

**[SWS\_CanTp\_00348]** [If `CanTpTxPaddingActivation` is equal to `CANTP_ON` for a Tx N-SDU, the CanTp module shall transmit by means of `CanIf_Transmit()` call, SF Rx N-PDU or last CF Rx N-PDU that belongs to that Tx N-SDU with the length of eight bytes (i.e. `PduInfoPtr.SduLength = 8`). Unused bytes in N-PDU shall be updated with `CANTP_PADDING_BYTE` (see [ECUC\\_CanTp\\_00298](#)). ] ( )

**[SWS\_CanTp\_00349]** [If `CanTpTxPaddingActivation` is equal to `CANTP_ON` for a Tx N-SDU, and if a FC N-PDU is received for that Tx N-SDU on a ongoing transmission, by means of `CanTp_RxIndication()` call, and the length of this FC is smaller than eight bytes (i.e. `PduInfoPtr.SduLength < 8`) the CanTp module shall abort the transmission session by calling `PduR_TxConfirmation()` with the result `E_NOT_OK`. If the development error detection is enabled, a development error `CANTP_E_PADDING` shall be triggered. ] ( )

### 7.3.9 Handling of unexpected N-PDU arrival

The behavior of the CAN Transport Layer on unexpected N-PDU arrival is greatly dependent on the communication direction type of the processing N-SDU.

**[SWS\_CanTp\_00057]** [If unexpected frames are received, the CanTp module shall behave according to the tables below. ] (SRS\_Can\_01082, SRS\_Can\_01117, SRS\_Can\_01149)

**[SWS\_CanTp\_00290]** [Those tables consider the actual CanTp internal status (CanTp status). Table 1 specifies the behavior on the half duplex implementation while table 2 defines the behavior for full duplex channels. ] (SRS\_Can\_01117, SRS\_Can\_01149)

It must be understood, that the received N-PDU contains the same address information (N\_AI) as the reception or transmission, which may be in progress at the time the N\_PDU is received.

<b>CanTp</b> status	<b>Reception of</b>				
	<b>SF N-PDU</b>	<b>FF N-PDU</b>	<b>CF N-PDU</b>	<b>FC N-PDU</b>	<b>Unknown N-PDU</b>
Segmented Transmit in progress	Ignore	Ignore	Ignore	If awaited, process the FC N-PDU, otherwise ignore it.	Ignore
Segmented Receive in progress	Terminate the current reception, report an indication, with parameter Result set to E_NOT_OK, to the upper layer, and process the SF N-PDU as the start of a new reception	Terminate the current reception, report an indication, with parameter Result set to E_NOT_OK, to the upper layer, and process the FF N-PDU as the start of a new reception	If awaited, process the CF N-PDU in the on-going reception and perform the required checks (e.g. SN in right order)	Ignore	Ignore
Idle <sup>1</sup>	Process the SF N-PDU as the start of a new reception	Process the FF N-PDU as the start of a new reception	Ignore	Ignore	Ignore

**Table 1: Handling of N-PDU arrivals for half duplex channels**

<sup>2</sup> Idle = CANTP\_ON.CANTP\_RX\_WAIT and CANTP\_ON.CANTP\_TX\_WAIT



<i>CanTp</i>	<i>Reception of</i>				
	<b>status</b>	<b>SF N-PDU</b>	<b>FF N-PDU</b>	<b>CF N-PDU</b>	<b>FC N-PDU</b>
Segmented Transmit in progress	If a reception is in progress process it according to the cell below, otherwise process the SF N-PDU as the start of a new reception	If a reception is in progress process it according to the cell below, otherwise process the FF N-PDU as the start of a new reception	If a reception is in progress process it according to the cell below, otherwise ignore it.	If awaited, process the FC N-PDU, otherwise ignore it.	Ignore
Segmented Receive in progress	Terminate the current reception, report an indication, with parameter Result set to E_NOT_OK, to the upper layer, and process the SF N-PDU as the start of a new reception	Terminate the current reception, report an indication, with parameter Result set to E_NOT_OK, to the upper layer, and process the FF N-PDU as the start of a new reception	Process the CF N-PDU in the ongoing reception and perform the required checks (e.g. SN in right order)	If a transmission is in progress process it according to the cell above, otherwise ignore it.	Ignore
Idle <sup>2</sup>	Process the SF N-PDU as the start of a new reception	Process the FF N-PDU as the start of a new reception	Ignore	Ignore	Ignore

**Table 2: Handling of N-PDU arrivals for full duplex channels**

## 7.4 Error classification

This section describes how the CanTp module has to manage the several error classes that may occur during the life cycle of this basic software.

The general requirements document of AUTOSAR [3] specifies that all basic software modules must distinguish (according to the product life cycle) two error types:

- Development errors: these errors should be detected and fixed during development phase. In most cases, these errors are software errors.
- Production errors: these errors are hardware errors and software exceptions that cannot be avoided and are expected to occur in the production (i.e. series) code.

<sup>2</sup> Idle = CANTP\_ON.CANTP\_RX\_WAIT and CANTP\_ON.CANTP\_TX\_WAIT

**[SWS\_CanTp\_00008]** [On errors and exceptions, the CanTp module shall not modify its current module state (see **Figure 4: CAN Transport Layer life cycle**) but shall simply report the error event. ] (SRS\_BSW\_00339)

**[SWS\_CanTp\_00291]** [In case of production error, the Diagnostic Event Manager module (via the Function Inhibition Manager) will perform the appropriate action (e.g. status modification of the calling module). ] ( )

**[SWS\_CanTp\_00293]** [The CanTp module shall be able to detect the following errors and exceptions depending on its configuration (development/production): ] ( )

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
API service called with wrong parameter(s): When CanTp_Transmit is called for a none configured PDU identifier or with an identifier for a received PDU.	Development	Could be a combination of: CANTP_E_PARAM_CONFIG CANTP_E_PARAM_ID	0x01 0x02
API service called with a NULL pointer. In case of this error, the API service shall return immediately without any further action, besides reporting this development error.	Development	CANTP_E_PARAM_POINTER	0x03
API service used without module initialization : On any API call except CanTp_Init() and CanTp_GetVersionInfo() if CanTp is in state "CANTP_OFF"	Development	CANTP_E_UNINIT	0x20
Invalid Transmit PDU identifier (e.g. a service is called with an inexistent Tx PDU identifier)	Development	CANTP_E_INVALID_TX_ID	0x30
Invalid Receive PDU identifier (e.g. a service is called with an inexistent Rx PDU identifier)	Development	CANTP_E_INVALID_RX_ID	0x40
PDU received with a length smaller than 8 bytes. (i.e. PduInfoPtr.SduLength < 8)	Development	CANTP_E_PADDING	0x70
CanTp_Transmit() is called for a configured Tx I-Pdu with functional addressing and the length parameter indicates, that the message can not be	Development	CANTP_E_INVALID_TATYPE	0x90

sent with a SF			
Requested operation is not supported – a cancel transmission/reception request for an N-SDU that it is not on transmission/reception process	Development	CANTP_E_OPER_NOT_SUPPORTED	0xA0
Another error occurred during a reception or a transmission: any protocol timeout error or implementation specific error	Development	CANTP_E_COM	0xB0
Event reported on completion of a reception operation	Development	CANTP_E_RX_COM	0xC0
Event reported on completion of a transmission operation	Development	CANTP_E_TX_COM	0xD0

## 7.5 Error detection

**[SWS\_CanTp\_00161]** [A static status variable, denoting whether a BSW module is initialized, should be initialized with value 0 before any APIs of the BSW module are called.

The initialization function of the BSW modules will set the static status variable to a value not equal to 0.

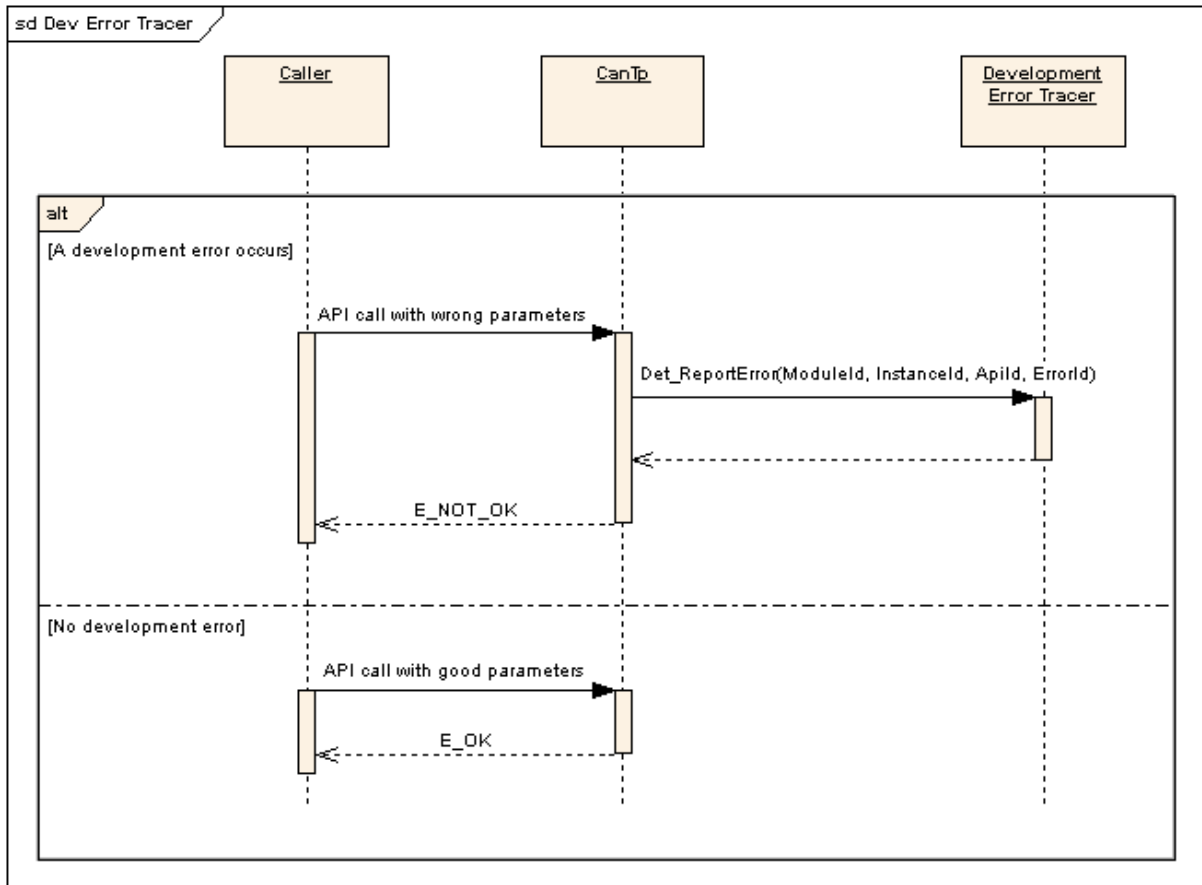
This variable is used to check if the module has been initialized before calling an API.

] (BSW406)

## 7.6 Error notification

**[SWS\_CanTp\_00115]** [The header file of the CanTp module, `CanTp.h`, shall provide a module ID, called `CANTP_MODULE_ID`, sets to the value 0x23. ] ( )

The following figure describes how this function can be used when the Development Error Tracer is on.



**Figure 11: Development error reporting**

**[SWS\_CanTp\_00294]** [As shown in the above figure, when a development error occurs the CanTp returns the value E\_NOT\_OK. The error description is only reported via the API of the Development Error Tracer module. ] ( )

**[SWS\_CanTp\_00229]** [If the task was aborted due to As, Bs, Cs, Ar, Br, Cr timeout, the CanTp module shall raise the DET error CANTP\_E\_RX\_COM (in case of a reception operation) or CANTP\_E\_TX\_COM (in case of a transmission operation). If the task was aborted due to any other protocol error, the CanTp module shall raise the DET error CANTP\_E\_COM. ] ( )

## 7.7 AUTOSAR debugging concept

For details refer to the chapter 7.1.17 “Debugging support” in *SWS\_BSWGeneral*.

## 8 API specification

### 8.1 Imported types

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

#### [SWS\_CanTp\_00209] [

<i>Module</i>	<i>Imported Type</i>
ComStack_Types	BufReq_ReturnType
	PduIdType
	PduLengthType
	RetryInfoType
	TPParameterType
	PduInfoType
Std_Types	Std_ReturnType
	Std_VersionInfoType

] ( )

In order to receive a consistent API for the AUTOSAR communication stack, basic types have been defined. These types are used by the CAN Transport Layer to communicate with the Pdu-Router and with the CAN Interface Layer.

For more information, these basic types are presented in depth in the AUTOSAR COM stack API specification.

These AUTOSAR standard types will be used without any type redefinition.

**[SWS\_CanTp\_00002]** [If, for implementation reasons, some additional types have to be defined, the CanTp module shall label these types as follows: CanTp\_<TypeName>Type, where <TypeName> is the name of this type adhering to the rules:

- No underscore usage
- First letter of each word upper case, consecutive letters lower case.

] (SRS\_BSW\_00353)

**[SWS\_CanTp\_00296]** [The CanTp module shall ensure that implementation-specific types are not "visible" outside of CanTp. Otherwise, the complete architecture would be corrupted. ] ( )

## 8.2 Type definitions

### 8.2.1 CanTp\_ConfigType

#### [SWS\_CanTp\_00340] [

<b>Name:</b>	CanTp_ConfigType
<b>Type:</b>	Structure
<b>Range:</b>	Implementation specific.
<b>Description:</b>	Data structure type for the post-build configuration parameters.

┌()

Implementation specific data structure type for the post-build configuration parameters.

### 8.3 Function definitions

This is a list of functions provided for upper layer modules

#### 8.3.1 CanTp\_Init

[SWS\_CanTp\_00208]

<b>Service name:</b>	CanTp_Init
<b>Syntax:</b>	void CanTp_Init( const CanTp_ConfigType* CfgPtr )
<b>Service ID[hex]:</b>	0x01
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	CfgPtr   Pointer to the CanTp post-build configuration data.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function initializes the CanTp module.

└ (SRS\_BSW\_00101, SRS\_BSW\_00358, SRS\_BSW\_00414)

After power up, CanTp is in a state called CANTP\_OFF (see [SWS\\_CanTp\\_00168](#)). In this state, the CanTp is not yet configured and therefore cannot perform any communication task.

The function CanTp\_Init initializes all global variables of the CAN Transport Layer with the given configuration set and set it in the idle state (state = CANTP\_ON but neither transmission nor reception are in progress) (see [SWS\\_CanTp\\_00170](#) and [SWS\\_CanTp\\_00030](#)).

The function CanTp\_Init has no return value because configuration data errors should be detected during configuration time (e.g. by the configuration tools). Furthermore, if a hardware error occurs, it will be reported via the error manager modules.

[SWS\_CanTp\_00199] [The CanTp module's environment shall call CanTp\_Init before using the CanTp module for further processing. ] ( )

[SWS\_CanTp\_00320] [If DET is enabled the function CanTp\_Init shall rise CANTP\_E\_PARAM\_POINTER error if the argument is a NULL pointer and return without any action. ] ( )

### 8.3.2 CanTp\_GetVersionInfo

[SWS\_CanTp\_00210]

<b>Service name:</b>	CanTp_GetVersionInfo
<b>Syntax:</b>	void CanTp_GetVersionInfo( Std_VersionInfoType* versioninfo )
<b>Service ID[hex]:</b>	0x07
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	versioninfo   Indicator as to where to store the version information of this module.
<b>Return value:</b>	None
<b>Description:</b>	This function returns the version information of the CanTp module.

] ( )

[SWS\_CanTp\_00319] [If DET is enabled the function CanTp\_GetVersionInfo shall rise CANTP\_E\_PARAM\_POINTER error if the argument is a NULL pointer and return without any action. ] ( )

### 8.3.3 CanTp\_Shutdown

[SWS\_CanTp\_00211]

<b>Service name:</b>	CanTp_Shutdown
<b>Syntax:</b>	void CanTp_Shutdown( void )
<b>Service ID[hex]:</b>	0x02
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	This function is called to shutdown the CanTp module.

] ( )

**[SWS\_CanTp\_00202]** [The function CanTp\_Shutdown shall close all pending transport protocol connections, free all resources and set the CanTp module into the CANTP\_OFF state. ] ( )

**[SWS\_CanTp\_00200]** [The function CanTp\_Shutdown shall not raise a notification about the pending frame transmission or reception. ] ( )

### 8.3.4 CanTp\_Transmit

**[SWS\_CanTp\_00212]**

<b>Service name:</b>	CanTp_Transmit	
<b>Syntax:</b>	Std_ReturnType CanTp_Transmit( PduIdType CanTpTxSduId, const PduInfoType* CanTpTxInfoPtr )	
<b>Service ID[hex]:</b>	0x03	
<b>Sync/Async:</b>	Asynchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	CanTpTxSduId	This parameter contains the unique CanTp module identifier of the CAN N-SDU to be transmitted. Range: 0..(maximum number of L-PDU IDs received) - 1
	CanTpTxInfoPtr	A pointer to a structure with CAN N-SDU related data: the length of the N-SDU that shall be transmitted, and a pointer to SDU data, which contains the addressing information for N-SDUs with MetaData (generic connections), and NULL otherwise.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: The request can be started successfully E_NOT_OK: The request cannot be started (e.g. a transmit request is in progress with the same N-SDU identifier)
<b>Description:</b>	This service is used to request the transfer of segmented data.	

] ( )



**[SWS\_CanTp\_00231]** [If data length is less than 7 or 6 (depending on normal or extended addressing format), the function `CanTp_Transmit` shall send a SF N-PDU. ] ( )

**[SWS\_CanTp\_00232]** [If data length is greater than 7 or 6 (depending on normal or extended addressing format), the function `CanTp_Transmit` shall initiate a multiple frame transmission session. ] ( )

**[SWS\_CanTp\_00204]** [The `CanTp` module shall notify the upper layer by calling the `PduR_CanTpTxConfirmation` callback when the transmit request has been completed. ] ( )

**[SWS\_CanTp\_00205]** [The `CanTp` module shall abort the transmit request and call the `PduR_CanTpTxConfirmation` callback function with the appropriate error result value if an error occurred (over flow, `N_A`s timeout, `N_B`s timeout and so on). ] ( )

The error result values are defined according to the ISO 15765 and definition of this type is depicted in the `ComStackTypes` document (AUTOSAR\_SWS\_ComStackTypes, Chapter 8.1.5).

**[SWS\_CanTp\_00206]** [The function `CanTp_Transmit` shall reject a request if the `CanTp_Transmit` service is called for a N-SDU identifier which is being used in a currently running CAN Transport Layer session. ] ( )

**[SWS\_CanTp\_00298]** [CanTp has limited buffering capability, and hence the N-SDU payload to be transmitted is not copied internally. The CAN Transport Layer obtains the data directly from the upper layer via the `PduR_CanTpCopyTxData` service. ] ( )

Thus, to guarantee the data consistency, the upper layer (e.g. DCM, `PduRouter` or AUTOSAR COM) must lock this memory area until the confirmation notification occurs.

**[SWS\_CanTp\_00299]** [When the upper layer calls this function for an N-SDU without `MetaData`, only the data length information of the structure indicated by `CanTpTxInfoPtr` has to be used. Its value indicates the payload length of the N-SDU, which is to be transmitted. For an N-SDU with `MetaData`, the data provided via the `CanTpTxInfoPtr` contains only `MetaData`. To obtain actual Tx data, the CAN Transport Layer shall call the `PduR_CanTpCopyTxData` service. ] ( )

**[SWS\_CanTp\_00321]** [If DET is enabled the function `CanTp_Transmit` shall rise `CANTP_E_PARAM_POINTER` error if the argument `CanTpTxInfoPtr` is a NULL pointer and return without any action. ] ( )

### 8.3.5 CanTp\_CancelTransmit

#### [SWS\_CanTp\_00246]

<b>Service name:</b>	CanTp_CancelTransmit	
<b>Syntax:</b>	Std_ReturnType CanTp_CancelTransmit ( PduIdType CanTpTxSduId )	
<b>Service ID[hex]:</b>	0x08	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CanTpTxSduId	This parameter contains the unique CanTp module identifier of the N-SDU to be cancelled for transmission. Range: 0..(maximum number of L-PDU IDs received) - 1
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Cancellation request of the specified N-SDU is accepted. E_NOT_OK: Cancellation request is rejected; the reason can be that request is issued for an N-SDU that is not segmented, request is issued after the last CF has been requested for transmission or cancellation is not possible for the related N-SDU due to configuration.
<b>Description:</b>	This service primitive is used to cancel the transfer of a pending CAN N-SDU. When the function returns, no transmission is in progress anymore with the given N-SDU identifier.	

] ( )

This service cancels the transmission of an N-SDU that has already requested for transmission by calling CanTp\_Transmit service.

**[SWS\_CanTp\_00254]** [If development error detection is enabled the function CanTp\_CancelTransmit shall check the validity of CanTpTxSduId parameter. If the parameter value is invalid, the CanTp\_CancelTransmit function shall raise the development error CANTP\_E\_PARAM\_ID and return E\_NOT\_OK.

If the parameter value indicates a cancel transmission request for an N-SDU that it is not on transmission process the CanTp module shall raise the DET error

CANTP\_E\_OPER\_NOT\_SUPPORTED and the service shall return E\_NOT\_OK. ] ( )

**[SWS\_CanTp\_00256]** [The CanTp shall abort the transmission of the current N-SDU if the service returns E\_OK. ] ( )

**[SWS\_CanTp\_00255]** [If the CanTp\_CancelTransmit service has been successfully executed the CanTp shall call the PduR\_CanTpTxConfirmation with notification result E\_NOT\_OK. ] ( )

### 8.3.6 CanTp\_CancelReceive

#### [SWS\_CanTp\_00257]

<b>Service name:</b>	CanTp_CancelReceive	
<b>Syntax:</b>	Std_ReturnType CanTp_CancelReceive ( PduIdType CanTpRxSduId )	
<b>Service ID[hex]:</b>	0x09	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	CanTpRxSduId	Identifier of the received N-SDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: Cancellation request of the specified N-SDU is accepted. E_NOT_OK: Cancellation request is rejected; the reason can be that request is issued for an N-SDU that is not segmented or request is issued for an N-SDU that is not in the reception process.
<b>Description:</b>	This service is used to cancel the reception of an ongoing N-SDU.	

The service CanTp\_CancelReceive cancels the reception of an N-SDU initiated by the reception of a First Frame and consequently calls of PduR\_StartOfReception. When the function returns, no reception is in progress anymore with the given N-SDU identifier ] ( )

**[SWS\_CanTp\_00260]** [If development error detection is enabled the function CanTp\_CancelReceive shall check the validity of CanTpRxSduId parameter. If the parameter value is invalid, the CanTp\_CancelReceive function shall raise the development error CANTP\_E\_PARAM\_ID and return E\_NOT\_OK. If the parameter value indicates a cancel reception request for an N-SDU that it is not on reception process the CanTp module shall raise the DET error CANTP\_E\_OPER\_NOT\_SUPPORTED and the service shall return E\_NOT\_OK. ] ( )

**[SWS\_CanTp\_00261]** [The CanTp shall abort the reception of the current N-SDU if the service returns E\_OK. ] ( )

**[SWS\_CanTp\_00262]** [The CanTp shall reject the request for receive cancellation in case of a Single Frame reception or if the CanTp is in the process of receiving the last Consecutive Frame of the N-SDU (i.e. the service is called after N-Cr timeout is started for the last Consecutive Frame). In this case the CanTp shall return E\_NOT\_OK. ] ( )

**[SWS\_CanTp\_00263]** [If the CanTp\_CancelReceive service has been successfully executed the CanTp shall call the PduR\_CanTpRxIndication with notification result E\_NOT\_OK. ] ( )

### 8.3.7 CanTp\_ChangeParameter

#### [SWS\_CanTp\_00302]

<b>Service name:</b>	CanTp_ChangeParameter	
<b>Syntax:</b>	Std_ReturnType CanTp_ChangeParameter ( PduIdType id, TPParameterType parameter, uint16 value )	
<b>Service ID[hex]:</b>	0x0a	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	id	Identifier of the received N-SDU on which the reception parameter has to be changed.
	parameter	Specify the parameter to which the value has to be changed (BS or STmin).
	value	The new value of the parameter.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	E_OK: request is accepted. E_NOT_OK: request is not accepted.
<b>Description:</b>	This service is used to request the change of reception parameters BS and STmin for a specified N-SDU.	

The service CanTp\_ChangeParameter is used to change the value of the reception parameter BS and STmin associated to each received N-SDU.

Implementation of this service depends on the configuration parameter *CanTpChangeParameterApi* (i.e. the service shall be implemented when the parameter is set to TRUE). ] ( )

**[SWS\_CanTp\_00303]** [A parameter change is only possible if the related N-SDU is not in the process of reception – i.e. a change of parameter value it is not possible after reception of FF until indication for last CF reception of the related N-SDU. ] ( )

**[SWS\_CanTp\_00304]** [If the change of a parameter is requested for an N-SDU that is on reception process the service CanTp\_ChangeParameter immediately returns E\_NOT\_OK and no parameter value is changed ] ( )

**[SWS\_CanTp\_00338]** [ When CanTp\_ChangeParameter is called for an N-SDU with MetaData (indicating a generic connection), the change shall be applied to all generic connections, so that it is used for all following receptions. ] ( )

**[SWS\_CanTp\_00305]** [If development error detection is enabled the function CanTp\_ChangeParameter shall check the validity of function parameters (Identifier, Parameter and requested value). If any of the parameter value is invalid, the

CanTp\_ChangeParameter function shall raise the development error CANTP\_E\_PARAM\_ID and return E\_NOT\_OK. | ( )

### 8.3.8 CanTp\_ReadParameter

#### [SWS\_CanTp\_00323]

<b>Service name:</b>	CanTp_ReadParameter	
<b>Syntax:</b>	Std_ReturnType CanTp_ReadParameter( PduIdType id, TPParameterType parameter, uint16* value )	
<b>Service ID[hex]:</b>	0x0b	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	id	Identifier of the received N-SDU on which the reception parameter are read.
	parameter	Specify the parameter to which the value has to be read (BS or STmin).
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	value	Pointer where the parameter value will be provided.
<b>Return value:</b>	Std_ReturnType	E_OK: request is accepted. E_NOT_OK: request is not accepted.
	<b>Description:</b> This service is used to read the current value of reception parameters BS and STmin for a specified N-SDU.	

Implementation of this service depends on the configuration parameter *CanTpReadParameterApi* (i.e. the service shall be implemented when the parameter is set to TRUE). | ( )

**[SWS\_CanTp\_00324]** [If development error detection is enabled the function CanTp\_ReadParameter shall check the validity of function parameters (Id and Parameter). If any of the parameter value is invalid, the CanTp\_ReadParameter function shall raise the development error CANTP\_E\_PARAM\_ID and return E\_NOT\_OK. | ( )

### 8.3.9 Main Function

#### [SWS\_CanTp\_00213]

<b>Service name:</b>	CanTp_MainFunction	
<b>Syntax:</b>	void CanTp_MainFunction( void )	
<b>Service ID[hex]:</b>	0x06	
<b>Description:</b>	The main function for scheduling the CAN TP.	

] ( )

**[SWS\_CanTp\_00164]** [The main function for scheduling the CAN TP (Entry point for scheduling)

The main function will be called by the Schedule Manager or by the Free Running Timer module according of the call period needed. CanTp\_MainFunction is involved in handling of CAN TP timeouts N\_As, N\_Bs, N-Cs, N\_Ar, N\_Br,

N\_Cr and STMmin. ] (SRS\_BSW\_00424, SRS\_BSW\_00373, SRS\_BSW\_00376)

**[SWS\_CanTp\_00300]** [The function CanTp\_MainFunction is affected by configuration parameter CanTpMainFunctionPeriod. ] ( )

## 8.4 Call-back notifications

The following is a list of functions provided for lower layer modules.

### 8.4.1 CanTp\_RxIndication

**[SWS\_CanTp\_00214]**

<b>Service name:</b>	CanTp_RxIndication	
<b>Syntax:</b>	<pre>void CanTp_RxIndication(     PduIdType RxPduId,     const PduInfoType* PduInfoPtr )</pre>	
<b>Service ID[hex]:</b>	0x42	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant for different Pdulds. Non reentrant for the same Pduld.	
<b>Parameters (in):</b>	RxPdulId	ID of the received I-PDU.
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	Indication of a received I-PDU from a lower layer communication interface module.	

] ( )

The CanIf module shall call this function after a successful reception of a Rx CAN L-PDU.

The data will be copied by the CanTp via the PDU structure PduInfoType. In this case the L-PDU buffers are not global and are therefore distributed in the corresponding CAN Transport Layer.

Note that PduInfoPtr contains also the MetaData in case of dynamic Rx N-PDUs.

**[SWS\_CanTp\_00235]** [The function CanTp\_RxIndication shall be callable in interrupt context (it could be called from the CAN receive interrupt). ] ( )

**[SWS\_CanTp\_00322]** [If DET is enabled the function CanTp\_RxIndication shall rise CANTP\_E\_PARAM\_POINTER error if the argument PduInfoPtr is a NULL pointer and return without any action. ] ( )

## 8.4.2 CanTp\_TxConfirmation

**[SWS\_CanTp\_00215]**

<b>Service name:</b>	CanTp_TxConfirmation
<b>Syntax:</b>	void CanTp_TxConfirmation( PduIdType TxPduId )
<b>Service ID[hex]:</b>	0x40
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Reentrant for different PduIds. Non reentrant for the same PduId.
<b>Parameters (in):</b>	TxPduId      ID of the I-PDU that has been transmitted.
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	The lower layer communication interface module confirms the transmission of an I-PDU.

] ( )

The CanIf module shall call the function CanTp\_TxConfirmation after the TP related CAN Frame (SF, FF, CF, FC) has been transmitted through the CAN network.

**[SWS\_CanTp\_00236]** [The function CanTp\_TxConfirmation shall be callable in interrupt context (it could be called from the CAN transmit interrupt). ] ( )

## 8.5 Expected Interfaces

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

### 8.5.1 Mandatory Interfaces

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

**[SWS\_CanTp\_00216]**

<b>API function</b>	<b>Description</b>
CanIf_Transmit	This service initiates a request for transmission of the CAN L-PDU specified by the CanTxSduId and CAN related data in the L-SDU structure.

PduR_CanTpRxIndication	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.
PduR_CanTpStartOfReception	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).
PduR_CanTpTxConfirmation	This function is called after the I-PDU has been transmitted via the TP API, the result indicates whether the transmission was successful or not.
PduR_CanTpCopyRxData	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 is written to the position indicated by bufferSizePtr.
PduR_CanTpCopyTxData	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.

] ( )

### 8.5.2 Optional Interfaces

This chapter defines the interface, which is required, in order to fulfill the optional functionality of the module.

#### [SWS\_CanTp\_00217]

<i>API function</i>	<i>Description</i>
Det_ReportError	Service to report development errors.

] ( )



## 9 Sequence diagrams

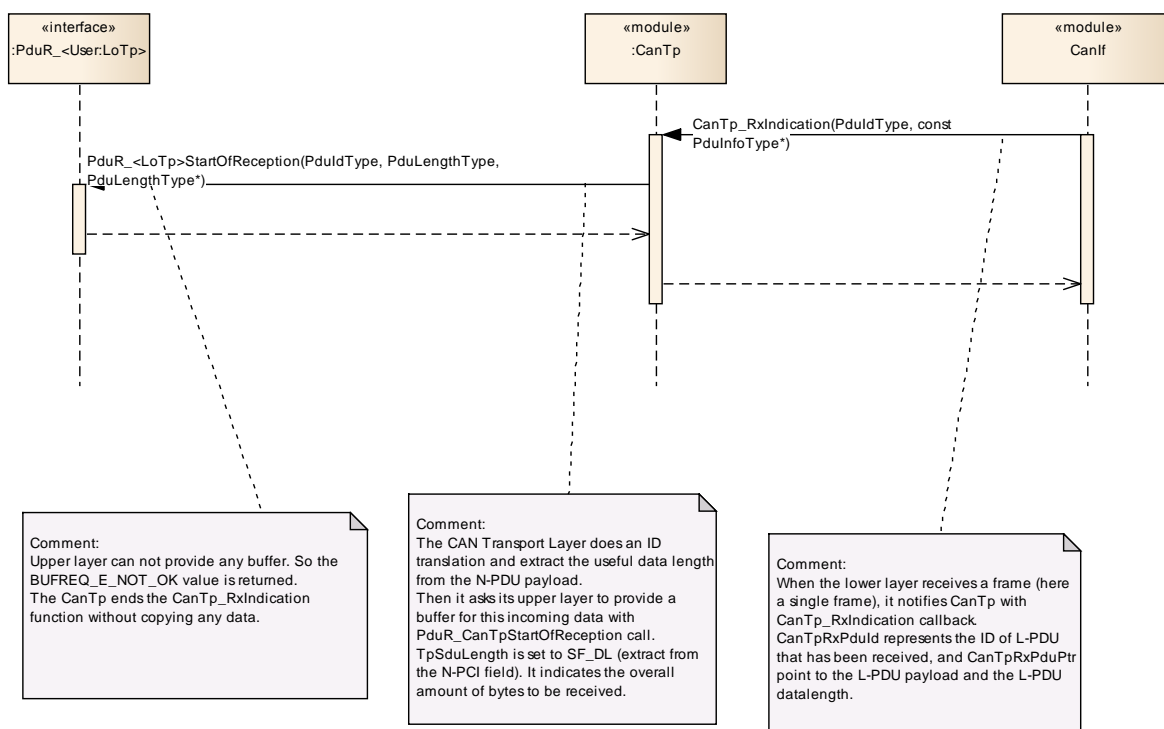
The goal of this chapter is to make it easier to understand the CAN Transport Layer by describing most of the more frequent and complicated use cases. Thus, the following diagram sequences are not exhaustive and do not reflect all the specified API possibilities.

### 9.1 SF N-SDU received and no buffer available.

#### 9.1.1 Assumptions

- All input parameters are OK;
- The N-SDU data length is smaller than or equal to 7 bytes (6 bytes in the case of extended or mixed addressing format);
- Upper layer can not make an Rx buffer available.

#### 9.1.2 Sequence diagram



**Note:** This sequence diagram demonstrates the working of the CAN\_Tp module only. However, if the whole system is considered during such reception, more modules are involved. Since this reception can be triggered in the context of CAN ISR, the CAN\_Tp operation should be as short as possible.

### 9.1.3 Transition description

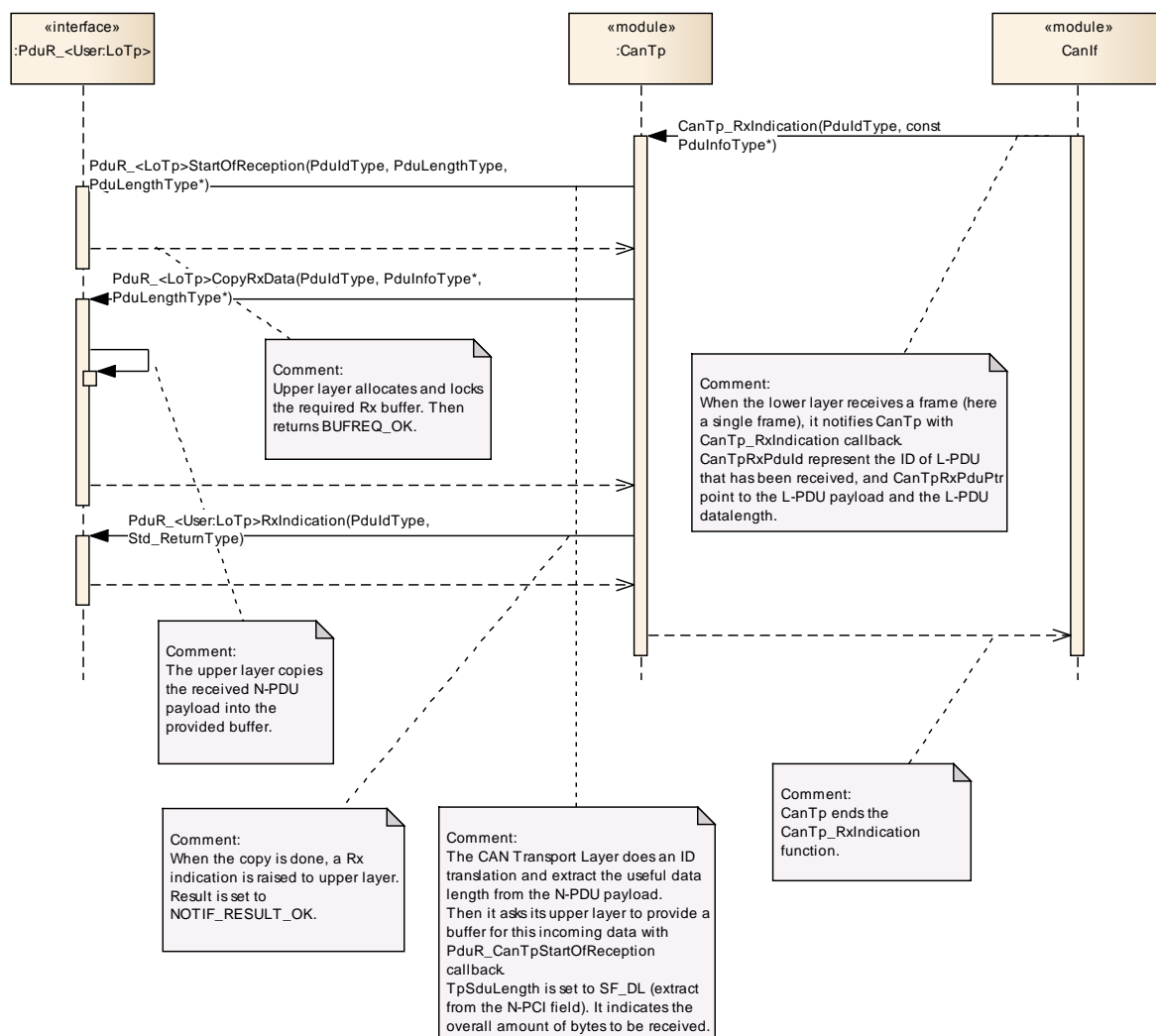
<b>Transition</b>	<b>Name</b>	<b>Description</b>
1	<pre>CanTp_RxIndication (     CanTpRxPduId,     CanTpRxPduPtr )</pre>	<p>When the lower layer receives a frame (here a single frame), it notifies CanTp by means of a CanTp_RxIndication callback. CanTpRxPduId represents the ID of L-PDU that has been received, and CanTpRxPduPtr indicates the L-PDU payload and the L-PDU data length.</p>
2	<pre>PduR_CanTpStartOfReception(     CanTpRxSduId,     TpSduLength,     PduInfoPtr )</pre>	<p>The CAN Transport Layer performs an ID translation and extracts the useful data length from the N-PDU payload. It then asks its upper layer to make a buffer available for this incoming data with a PduR_CanTpStartOfReception callback.</p> <p>TpSduLength is set to SF_DL (extracted from the N-PCI field). It indicates the overall amount of bytes to be received.</p>
3	<pre>BUFREQ_E_NOT_OK</pre>	<p>The upper layer cannot make any buffer available, so the BUFREQ_E_NOT_OK value is returned.</p> <p>The CanTp ends the CanTp_RxIndication function without copying any data.</p>

## 9.2 Successful SF N-PDU reception

### 9.2.1 Assumptions

- All input parameters are OK;
- The N-SDU data length is smaller than or equal to 7 bytes (6 bytes in the case of extended addressing format);
- The SF N-PDU is successfully received.

### 9.2.2 Sequence diagram



**Note:** This sequence diagram demonstrates the working of the CAN\_Tp module only. However, if the whole system is considered during such reception, more modules are involved. Since this reception can be triggered in the context of CAN ISR, the CAN\_Tp operation should be as short as possible.

### 9.2.3 Transition description

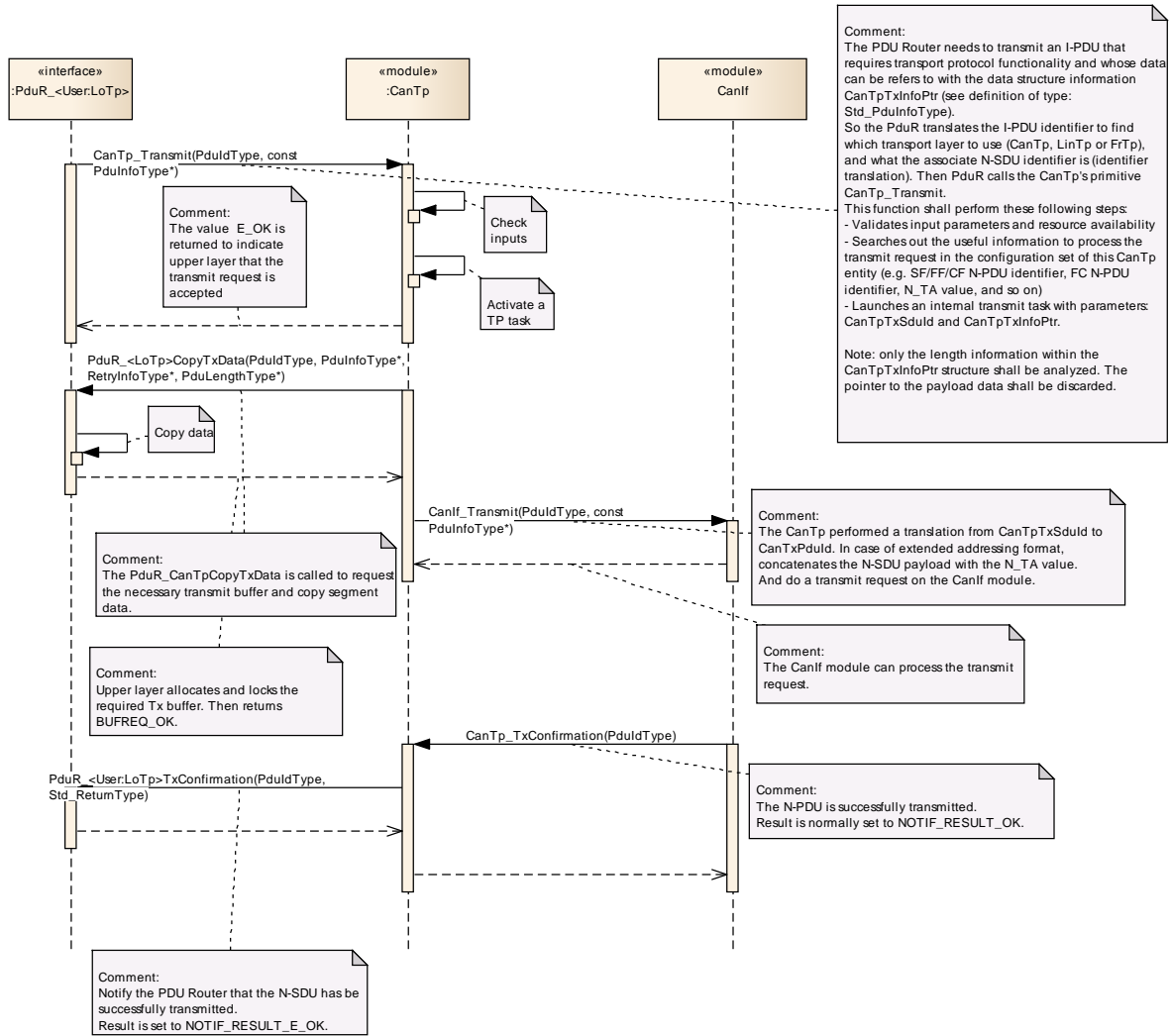
<b>Transition</b>	<b>Name</b>	<b>Description</b>
1	CanTp_RxIndication ( CanTpRxPduId, CanTpRxPduPtr )	When the lower layer receives a frame (here a single frame), it notifies CanTp by means of a CanTp_RxIndication callback. CanTpRxPduId represents the ID of the L-PDU that has been received, and CanTpRxPduPtr indicates the L-PDU payload and the L-PDU data length.
2	PduR_CanTpStartOfReception( CanTpRxSduId, TpSduLength, PduInfoPtr )	The CAN Transport Layer performs an ID translation and extracts the useful data length from the N-PDU payload. It then asks its upper layer to make a buffer available for this incoming data with a PduR_CanTpStartOfReception callback. TpSduLength is set to SF_DL (extracted from the N-PCI field). It indicates the overall amount of bytes to be received.
3	BUFREQ_OK	Upper layer allocates and locks the required Rx buffer. Then returns BUFREQ_OK.
4	PduR_CanTpCopyRxData( PduIdType, PduInfoType*, PduLengthType )	The upper layer copies the received N-PDU payload into the buffer.
5	PduR_CanTpRxIndication( CanTpRxSduId, Result )	When the copy is complete, an Rx indication is sent to the upper layer. The result is set to E_OK.
6		CanTp ends the CanTp_RxIndication function.

## 9.3 Transmit request of SF N-SDU

### 9.3.1 Assumptions

- All input parameters are OK;
- The N-SDU data length is smaller than or equal to 7 bytes (6 bytes in case of extended addressing format);
- The transmission is successfully processed.

9.3.2 Sequence diagram



### 9.3.3 Transition description

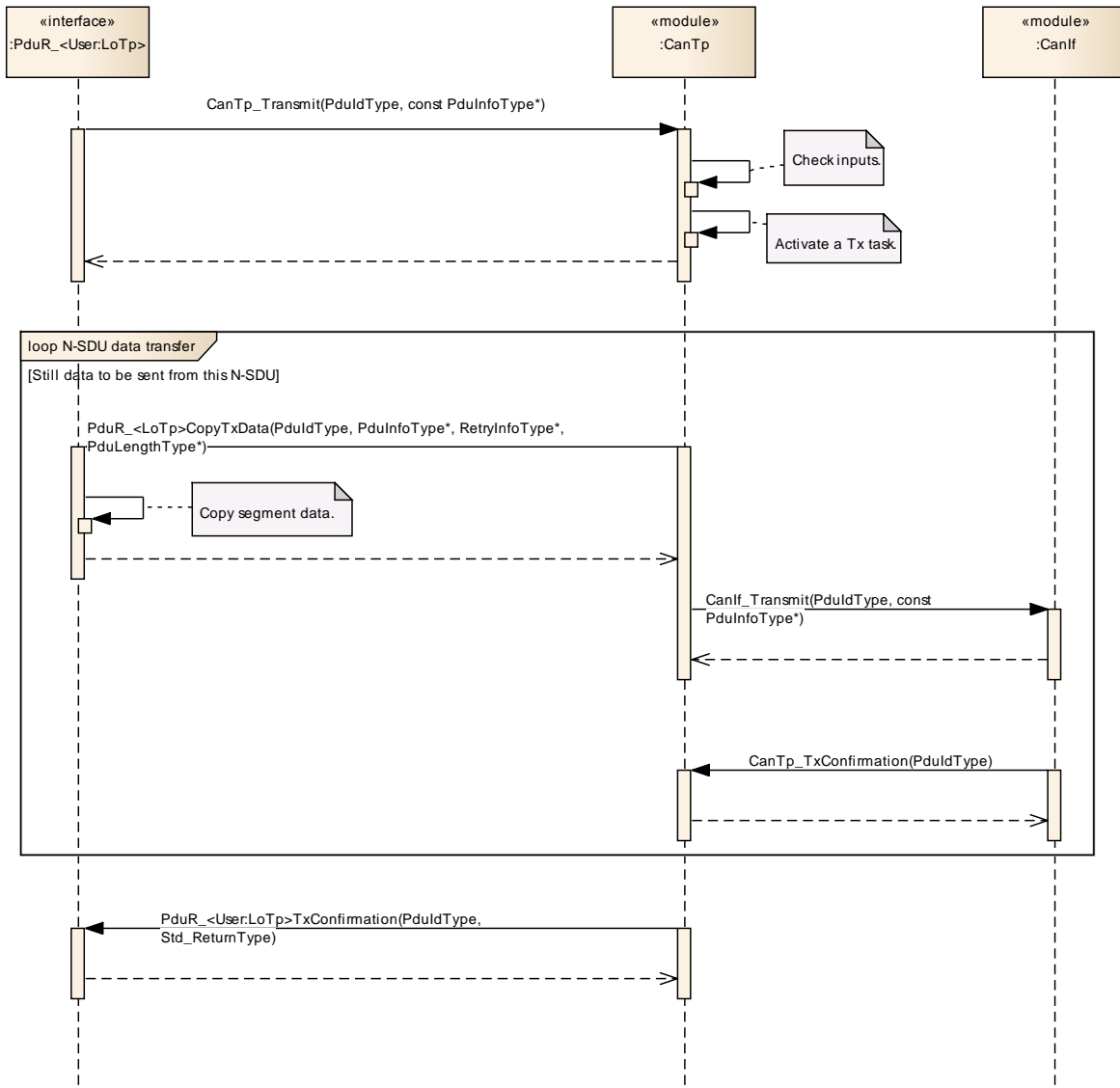
<b>Transition</b>	<b>Name</b>	<b>Description</b>
1	<pre>CanTp_Transmit(   CanTpTxSduId,   CanTpTxInfoPtr )</pre>	<p>The PDU Router needs to transmit an I-PDU that requires transport protocol functionality and whose data can be refers to with the data structure information CanTpTxInfoPtr (see definition of type: Std_PduInfoType). So the PduR translates the I-PDU identifier to find which transport layer to use (CanTp, LinTp or FrTp), and what the associate N-SDU identifier is (identifier translation). Then PduR calls the CanTp's primitive CanTp_Transmit. This function shall perform these following steps:</p> <ul style="list-style-type: none"> <li>- Validates input parameters and resource availability</li> <li>- Searches out the useful information to process the transmit request in the configuration set of this CanTp entity (e.g. SF/FF/CF N-PDU identifier, FC N-PDU identifier, N_TA value, and so on)</li> <li>- Launches an internal transmit task with parameters: CanTpTxSdulId and CanTpTxInfoPtr.</li> </ul> <p>Note: only the length information within the CanTpTxInfoPtr structure shall be analyzed. The pointer to the payload data shall be discarded.</p>
2	E_OK	<p>The value E_OK is returned to indicate to the upper layer that the transmit request is accepted. The upper layer locks the required Tx buffer.</p>
3	<pre>PduR_CanTpCopyTxdata(   SduId,   PduInfoPtr,   TxState,   TxCntPtr )</pre>	<p>The PduR_CanTpCopyTxData is called to copy segment data.</p>
4	BUFREQ_OK	Upper layer copy data, then returns BUFREQ_E_OK.
5	<pre>CanIf_Transmit(   CanTxPduId,   PduInfoPtr )</pre>	<p>The CanTp performs a translation from CanTpTxSdulId to CanTxPdulId. In case of extended addressing format, it concatenates the N-SDU payload with the N_TA value, to perform a transmit request on the CanIf module.</p>
6	E_OK	The CanIf module can process the transmit request.
7	<pre>CanTp_TxConfirmation(   CanTpTxPduId, )</pre>	<p>The N-PDU is successfully transmitted.</p>
8	<pre>PduR_CanTpTxConfirmation(   CanTpTxSduId,   Result )</pre>	<p>Notifies the PDU Router that the N-SDU has been successfully transmitted. Consequently, the PduInfoType structure has to be unlocked. Result is set to E_OK.</p>

## 9.4 Transmit request of larger N-SDU

### 9.4.1 Assumptions

- All input parameters are OK;
- The N-SDU data length is larger than 7 bytes (6 bytes in case of extended or mixed addressing format);
- The transmission is successfully processed.

**9.4.2 Sequence diagram**





### 9.4.3 Transition description

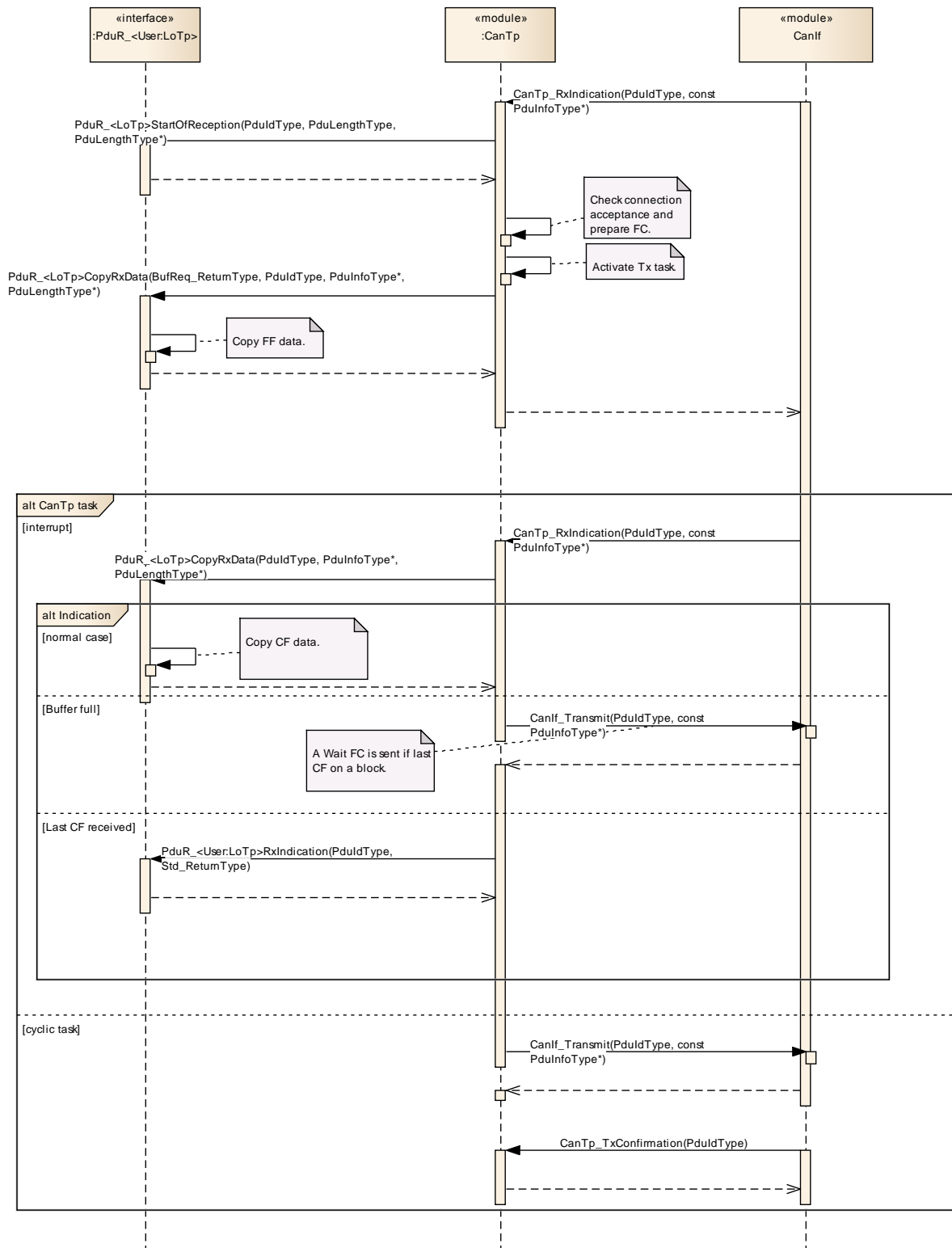
<b>Transition</b>	<b>Name</b>	<b>Description</b>
1	<pre>CanTp_Transmit (   CanTpTxSduId,   CanTpTxInfoPtr )</pre>	<p>The PDU Router needs to transmit an I-PDU that requires transport protocol functionality and whose data refers to with the data structure information CanTpTxInfoPtr (see definition of type: PduInfoType).</p> <p>So the PduR translates the I-PDU identifier to find which transport layer to use (CanTp, LinTp or FrTp), and what the associate N-SDU identifier is (identifier translation). Then PduR calls the CanTp's primitive CanTp_Transmit. This function shall perform these following steps:</p> <ul style="list-style-type: none"> <li>- Validates input parameters and resource availability</li> <li>- Searches out the useful information to process the transmit request in the configuration set of this CanTp entity (e.g. SF/FF/CF N-PDU identifier, FC N-PDU identifier, N_TA value, and so on)</li> <li>- Launches an internal transmit task with parameters: CanTpTxSdulId and CanTpTxInfoPtr.</li> </ul> <p>Note: only the length information within the CanTpTxInfoPtr structure shall be analyzed. The pointer to the payload data shall be discarded.</p> <p>Upon successful return of the call, the upper layer locks the required Tx buffer.</p>
2	E_OK	The upper layer allocates and locks the required Tx buffer. Then returns E_OK.
3	<pre>PduR_CanTpCopyTxData (   SduId,   PduInfoPtr,   TxState,   TxCntPtr )</pre>	The PduR_CanTpCopyTxData is called. The upper layer copies segment data into the destination buffer.
4	<pre>CanIf_Transmit (   CanTxPduId,   PduInfoPtr )</pre>	Within the task, CanTp calls the CAN Interface by using CanIf_Transmit, where CanTxPduId identifies the L-SDU (a translation has to be preformed between the N-SDU Id used by CanTp and the L-SDU Id used by CAN Interface), and PduInfoPtr indicator data and their length.
5	<pre>CanTp_TxConfirmation (   CanTpTxPduId, )</pre>	CanTp awaits a confirmation from the CAN Interface (CanTp_TxConfirmation)
6	<pre>PduR_CanTpCopyTxData (   SduId,   PduInfoPtr,   TxState,   TxCntPtr )</pre>	For each consecutive frame CanTp asks the PDU Router to provide new data to be sent.
7	<pre>PduR_CanTpTxConfirmation (   CanTpTxSduId,   Result )</pre>	When all data have been sent, or when an error occurs, CanTp notifies PDU Router with PduR_CanTpTxConfirmation. CanTpTxPduId identify the N-SDU which transmission is confirmed, and result indicates if transmission has been completed or not.

## 9.5 Large N-SDU Reception

### 9.5.1 Assumptions

- All input parameters are OK;
- The N-SDU data length is larger than 7 bytes (6 bytes in case of extended addressing format);
- Reception is successfully processed.

**9.5.2 Sequence diagram**



**Note :** This sequence diagram demonstrates the working of the CAN\_Tp module only. However, if the whole system is considered in such reception, more modules are involved. Since this reception can be triggered in the context of a CAN ISR, the CAN Tp operation should be as short as possible.

### 9.5.3 Transition description

<b>Transition</b>	<b>Name</b>	<b>Description</b>
1	CanTp_RxIndication ( CanTpRxPduId, CanTpRxPduPtr )	When the CAN Interface receives a frame (here a first frame), it notifies CanTp by means of a CanTp_RxIndication callback. CanTpRxPduId represents the ID of L-PDU that has been received and CanTpRxPduPtr indicates payload and L-SDU datalength to the L-SDU.
2	PduR_CanTpStartOfReception( CanTpRxSduId, TpSduLength, PduInfoPtr )	CanTp ask PDU Router to make a buffer available for incoming data with PduR_CanTpStartOfReception callback.
3		Check connection acceptance and prepare FC parameters.
4		CanTp activates a task for sending an FC with a Flow Status set to ContinueToSend. (see step 8.)
5	CanTp_RxIndication ( CanTpRxPduId, CanTpRxPduPtr )	When the CAN Interface receives a frame (here a consecutive frame), CAN Interface notifies CanTp by means of a CanTp_RxIndication callback. CanTpRxPduId represents the ID of the CAN frame that has been received and CanTpRxPduPtr indicates payload to the L-SDU.
6		CanTp shall verify the sequence number and if correct, it asks the PduR to copy the data.
7	PduR_CanTpCopyRxData( PduIdType, PduInfoType*, PduLengthType )  Or  PduR_CanTpRxIndication( CanTpRxSduId, Result )	Three cases can apply :  - Normal Case: the received consecutive frame is not the last one. CanTp forwards received data to the upper layer.  - Last CF Received: this consecutive frame is the last (Total length information was, as parameter, in the first frame). CanTp shall notify PDU Router with PduR_CanTpRxIndication callback.
8		When flow control needs to be sent, the CanTp cyclical task should call the CAN Interface by using CanIf_Transmit and wait confirmation from the CAN Interface.  Flow control can be either ContinueToSend or Wait, depending on the available buffer in the upper layer.

## 10 Configuration specification

This chapter defines configuration parameters and their clustering into containers. In order to support the specification, Chapter 10.1 describes fundamentals.

Chapter 10.2 specifies the structure (containers) and the parameters of the module CAN Transport Layer.

Chapter 10.3 specifies published information for the module CAN Transport Layer

**[SWS\_CanTp\_00146]** [The listed configuration items can be derived from a network description database, which is based on the EcuConfigurationTemplate. The configuration tool should extract all information to configure the CAN Transport Protocol. ] (SRS\_BSW\_00159)

**[SWS\_CanTp\_00147]** [The consistency of the configuration must be checked by the configuration tool at configuration time. ] (SRS\_BSW\_00167)

### 10.1 How to read this chapter

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

## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in Chapters 7 and 8.

**[SWS\_CanTp\_00328]** [The same NPdu may only be referenced by more than one NSdu (RxNSdu or TxNSdu, via CanTpRxNPdu, CanTpTxFcNPdu, CanTpTxNPdu, or CanTpRxFcNPdu), when either the NSdu has addressing format extended or mixed (29 or 11 bit), or when the NPdu has MetaData. ] ()

### 10.2.1 Variants

For details refer to the chapter 10.1.2 “Variants” in *SWS\_BSWGeneral*.

### 10.2.2 CanTp

<b>Module Name</b>	<i>CanTp</i>
<b>Module Description</b>	Configuration of the CanTp (CAN Transport Protocol) module.

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanTpConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR CanTp module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
CanTpGeneral	1	This container contains the general configuration parameters of the CanTp module.

### 10.2.3 CanTpConfig

<b>SWS Item</b>	<b>ECUC_CanTp_00290 :</b>
<b>Container Name</b>	CanTpConfig [Multi Config Container]
<b>Description</b>	This container contains the configuration parameters and sub containers of the AUTOSAR CanTp module. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CanTp_00240 :</b>		
<b>Name</b>	CanTpMainFunctionPeriod {CANTP_MAIN_FUNCTION_PERIOD}		
<b>Description</b>	Allow to configure the time for the MainFunction (as float in seconds). The CanTpMainFunctionPeriod should be assigned a value which is optimal regarding all of the timers configured for CanTp in TX and RX data transfer i.e. the differences from the configured timing should be as small as possible. Please note: This period shall be the same as call cycle time of the periodic task were CanTp Main function is called.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. 0.255		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	

<b>Scope / Dependency</b>	scope: ECU
---------------------------	------------

<b>SWS Item</b>	<b>ECUC_CanTp_00304 :</b>		
<b>Name</b>	CanTpMaxChannelCnt		
<b>Description</b>	Maximum number of channels. This parameter is needed only in case of post-build loadable implementation using static memory allocation.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 ..		
	18446744073709551615		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanTpChannel	1..*	This container contains the configuration parameters of the CanTp channel.

### 10.2.4 CanTpGeneral

<b>SWS Item</b>	<b>ECUC_CanTp_00278 :</b>		
<b>Container Name</b>	CanTpGeneral{CanTpConfiguration}		
<b>Description</b>	This container contains the general configuration parameters of the CanTp module.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00299 :</b>		
<b>Name</b>	CanTpChangeParameterApi {CANTP_CHANGE_PARAMETER_API}		
<b>Description</b>	This parameter, if set to true, enables the CanTp_ChangeParameterRequest Api for this Module.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00239 :</b>		
<b>Name</b>	CanTpDevErrorDetect {CANTP_DEV_ERROR_DETECT}		
<b>Description</b>	Switches the Development Error Detection and Notification ON or OFF		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00302 :</b>		
<b>Name</b>	CanTpDynIdSupport {CANTP_DYNID_SUPPORT}		
<b>Description</b>	Enable support for dynamic ID handling via N-PDU MetaData.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		

<b>Default value</b>	false		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00303 :</b>		
<b>Name</b>	CanTpGenericConnectionSupport {CANTP_GENERIC_CONNECTION_SUPPORT}		
<b>Description</b>	Enable support for the handling of generic connections using N-SDUs with MetaData. Requires CanTpDynIdSupport.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local dependency: Requires CanTpDynIdSupport.		

<b>SWS Item</b>	<b>ECUC_CanTp_00298 :</b>		
<b>Name</b>	CanTpPaddingByte {CANTP_PADDING_BYTE}		
<b>Description</b>	Used for the initialization of unused bytes with a certain value		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00300 :</b>		
<b>Name</b>	CanTpReadParameterApi {CANTP_READ_PARAMETER_API}		
<b>Description</b>	This parameter, if set to true, enables the CanTp_ReadParameterApi for this module.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00283 :</b>		
<b>Name</b>	CanTpVersionInfoApi		
<b>Description</b>	The function CanTp_GetVersionInfo is configurable (On/Off) by this configuration parameter.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			



**No Included Containers**

### 10.2.5 CanTpChannel

<b>SWS Item</b>	<b>ECUC_CanTp_00288 :</b>
<b>Container Name</b>	CanTpChannel
<b>Description</b>	This container contains the configuration parameters of the CanTp channel.  <b>Attributes:</b> postBuildChangeable=true
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CanTp_00289 :</b>	
<b>Name</b>	CanTpChannelMode	
<b>Description</b>	The CAN Transport Layer supports half and full duplex channel modes.	
<b>Multiplicity</b>	1	
<b>Type</b>	EcucEnumerationParamDef	
<b>Range</b>	CANTP_MODE_FULL_DUPLEX	Full duplex channel.
	CANTP_MODE_HALF_DUPLEX	Half duplex channel.
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X All Variants
	<b>Link time</b>	--
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>	scope: local	

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanTpRxNSdu	0..*	The following parameters needs to be configured for each CAN N-SDU that the CanTp module receives via the CanTpChannel.
CanTpTxNSdu	0..*	The following parameters needs to be configured for each CAN N-SDU that the CanTp module transmits via the CanTpChannel.

### 10.2.6 CanTpRxNSdu

<b>SWS Item</b>	<b>ECUC_CanTp_00137 :</b>	
<b>Container Name</b>	CanTpRxNSdu{RxNsdu}	
<b>Description</b>	The following parameters needs to be configured for each CAN N-SDU that the CanTp module receives via the CanTpChannel.  <b>Attributes:</b> postBuildChangeable=true	
<b>Configuration Parameters</b>		

<b>SWS Item</b>	<b>ECUC_CanTp_00276 :</b>	
<b>Name</b>	CanTpBs {CANTP_BS}	
<b>Description</b>	Sets the number of N-PDUs the CanTp receiver allows the sender to send, before waiting for an authorization to continue transmission of the following N-PDUs.For further details on this parameter value see ISO 15765-2 specification.	
<b>Multiplicity</b>	0..1	
<b>Type</b>	EcucIntegerParamDef	
<b>Range</b>	0 .. 255	
<b>Default value</b>	--	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X VARIANT-PRE-COMPILE

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

<b>SWS Item</b>	<b>ECUC_CanTp_00277 :</b>		
<b>Name</b>	CanTpNar {CANTP_NAR}		
<b>Description</b>	Value in seconds of the N_Ar timeout. N_Ar is the time for transmission of a CAN frame (any N_PDU) on the receiver side.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00245 :</b>		
<b>Name</b>	CanTpNbr {CANTP_NBR}		
<b>Description</b>	Value in seconds of the performance requirement for (N_Br + N_Ar). N_Br is the elapsed time between the receiving indication of a FF or CF or the transmit confirmation of a FC, until the transmit request of the next FC.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00279 :</b>		
<b>Name</b>	CanTpNcr {CANTP_NCR}		
<b>Description</b>	Value in seconds of the N_Cr timeout. N_Cr is the time until reception of the next Consecutive Frame N_PDU.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00281 :</b>		
<b>Name</b>	CanTpRxAddressingFormat {CANTP_RX_ADDRESSING_FORMAT}		
<b>Description</b>	Declares which communication addressing mode is supported for this RxNSdu. Definition of Enumeration values: CanTpStandard to use normal addressing format. CanTpExtended to use extended addressing format. CanTpMixed to use mixed 11 bit addressing format. CanTpNormalFixed to use normal fixed addressing format. CanTpMixed29Bit to use mixed 29 bit addressing format.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	CANTP_EXTENDED		Extended addressing format
	CANTP_MIXED		Mixed 11 bit addressing format

	CANTP_MIXED29BIT	Mixed 29 bit addressing format	
	CANTP_NORMALFIXED	Normal fixed addressing format	
	CANTP_STANDARD	Normal addressing format	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00280 : (Obsolete)</b>		
<b>Name</b>	CanTpRxDI {CANTP_DL}		
<b>Description</b>	<p>Please note that this parameter is deprecated and will be removed in a future release.</p> <p>Old description: Data Length Code of this RxNsdU. In case of variable message length, this value indicates the minimum data length. Depending on SF or FF N-SDU the value will be limited to 7 (6 for an extended addressing format) and 4095 respectively.</p> <p><b>Tags:</b> atp.Status=obsolete atp.StatusRevisionBegin=4.1.1</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00301 :</b>		
<b>Name</b>	CanTpRxNSduId {CANTP_RXNSDU_ID}		
<b>Description</b>	Unique identifier user by the upper layer to call CanTp_CancelReceive, CanTp_ChangeParameter and CanTp_ReadParameter.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00249 :</b>		
<b>Name</b>	CanTpRxPaddingActivation {CANTP_PADDING_ACTIVATION}		
<b>Description</b>	<p>Defines if the receive frame uses padding or not.</p> <p>Definition of enumeration values: CanTpOn: The N-PDU received uses padding for SF, FC and the last CF. (N-PDU length is always 8 bytes) CanTpOff: The N-PDU received does not use padding for SF, CF and the last CF. (N-PDU length is dynamic)</p>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	CANTP_OFF	Padding is not used	
	CANTP_ON	Padding is used	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00250 :</b>		
<b>Name</b>	CanTpRxTaType {CANTP_TA_TYPE}		
<b>Description</b>	Declares the communication type of this Rx N-SDU.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	CANTP_FUNCTIONAL	Functional request type	
	CANTP_PHYSICAL	Physical request type	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00251 :</b>		
<b>Name</b>	CanTpRxWftMax {CANTP_WFTMAX}		
<b>Description</b>	<p>This parameter indicates how many Flow Control wait N-PDUs can be consecutively transmitted by the receiver. It is local to the node and is not transmitted inside the FC protocol data unit.</p> <p>CanTpRxWftMax is used to avoid sender nodes being potentially hooked-up in case of a temporarily reception inability on the part of the receiver nodes, whereby the sender could be waiting continuously.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00252 :</b>		
<b>Name</b>	CanTpSTmin {CANTP_STMIN}		
<b>Description</b>	<p>Sets the duration of the minimum time the CanTp sender shall wait between the transmissions of two CF N-PDUs.</p> <p>For further details on this parameter value see ISO 15765-2 specification.</p>		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00241 :</b>		
<b>Name</b>	CanTpRxNSduRef		
<b>Description</b>	Reference to a Pdu in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>

CanTpNAe	0..1	This container is required for each RxNSdu and TxNSdu with AddressingFormat CANTP_MIXED or CANTP_MIXED29BIT.
CanTpNSa	0..1	This container is required for each RxNSdu and TxNSdu with RxTaType CANTP_PHYSICAL and CanTpAddressingFormat CANTP_EXTENDED. When DynIdSupport is enabled, this container is also required for each TxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT. When DynIdSupport is enabled and GenericConnectionSupport is not enabled, this container is also required for each RxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT.
CanTpNTa	0..1	This container is required for each RxNSdu and TxNSdu with AddressingFormat CANTP_EXTENDED. When DynIdSupport is enabled, this container is also required for each RxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT. When DynIdSupport is enabled and GenericConnectionSupport is not enabled, this container is also required for each TxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT.
CanTpRxNPdu	1	Used for grouping of the ID of a PDU and the Reference to a PDU.
CanTpTxFcNPdu	0..1	Used for grouping of the ID of a PDU and the Reference to a PDU.

### 10.2.7 CanTpRxNPdu

<b>SWS Item</b>	<b>ECUC_CanTp_00256 :</b>		
<b>Container Name</b>	CanTpRxNPdu		
<b>Description</b>	Used for grouping of the ID of a PDU and the Reference to a PDU.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00258 :</b>		
<b>Name</b>	CanTpRxNPduId {CANTP_RXNPDU_ID}		
<b>Description</b>	The N-PDU identifier attached to the RxNsdu is identified by CanTpRxNSduId. Each RxNsdu identifier is linked to only one SF/FF/CF N-PDU identifier. Nevertheless, in the case of extended or mixed addressing format, the same N-PDU identifier can be used for several N-SDU identifiers. The distinction is made by the N_TA or N_AE value (first data byte of SF or FF frames).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00257 :</b>		
<b>Name</b>	CanTpRxNPduRef		
<b>Description</b>	Reference to a Pdu in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

No Included Containers

### 10.2.8 CanTpTxFcNPdu

<b>SWS Item</b>	<b>ECUC_CanTp_00259 :</b>		
<b>Container Name</b>	CanTpTxFcNPdu		
<b>Description</b>	Used for grouping of the ID of a PDU and the Reference to a PDU.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00287 :</b>		
<b>Name</b>	CanTpTxFcNPduConfirmationPduld		
<b>Description</b>	Handle Id to be used by the CanIf to confirm the transmission of the CanTpTxFcNPdu to the CanIf module.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00260 :</b>		
<b>Name</b>	CanTpTxFcNPduRef		
<b>Description</b>	Reference to a Pdu in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

No Included Containers

### 10.2.9 CanTpTxNSdu

<b>SWS Item</b>	<b>ECUC_CanTp_00138 :</b>		
<b>Container Name</b>	CanTpTxNSdu{TxNsdu}		
<b>Description</b>	The following parameters needs to be configured for each CAN N-SDU that the CanTp module transmits via the CanTpChannel.  <b>Attributes:</b> postBuildChangeable=true		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00263 :</b>		
<b>Name</b>	CanTpNas {CANTP_NAS}		
<b>Description</b>	Value in second of the N_As timeout. N_As is the time for transmission of a CAN frame (any N_PDU) on the part of the sender.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD

<b>Scope / Dependency</b>	scope: local
---------------------------	--------------

<b>SWS Item</b>	<b>ECUC_CanTp_00264 :</b>		
<b>Name</b>	CanTpNbs {CANTP_NBS}		
<b>Description</b>	Value in seconds of the N_Bs timeout. N_Bs is the time of transmission until reception of the next Flow Control N_PDU.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00265 :</b>		
<b>Name</b>	CanTpNcs {CANTP_NCS}		
<b>Description</b>	Value in seconds of the performance requirement of (N_Cs + N_As). N_Cs is the time which elapses between the transmit request of a CF N-PDU until the transmit request of the next CF N-PDU.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	0 .. INF		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00282 :</b>		
<b>Name</b>	CanTpTc {CANTP_TC}		
<b>Description</b>	Switch for enabling Transmit Cancellation and Receive Cancellation.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00262 :</b>		
<b>Name</b>	CanTpTxAddressingFormat {CANTP_TX_ADDRESSING_FORMAT}		
<b>Description</b>	Declares which communication addressing format is supported for this TxNSdu. Definition of Enumeration values: CanTpStandard to use normal addressing format. CanTpExtended to use extended addressing format. CanTpMixed to use mixed 11 bit addressing format. CanTpNormalFixed to use normal fixed addressing format. CanTpMixed29Bit to use mixed 29 bit addressing format.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	CANTP_EXTENDED	Extended addressing format	
	CANTP_MIXED	Mixed 11 bit addressing format	
	CANTP_MIXED29BIT	Mixed 29 bit addressing format	
	CANTP_NORMALFIXED	Normal fixed addressing format	
	CANTP_STANDARD	Normal addressing format	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants

	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00267 : (Obsolete)</b>		
<b>Name</b>	CanTpTxDI {CANTP_DL}		
<b>Description</b>	Please note that this parameter is deprecated and will be removed in a future release. Old description: Data Length Code of this TxNsdu. In case of variable length message, this value indicates the minimum data length. <b>Tags:</b> atp.Status=obsolete atp.StatusRevisionBegin=4.1.1		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00268 :</b>		
<b>Name</b>	CanTpTxNSduId {CANTP_TXNSDU_ID}		
<b>Description</b>	Unique identifier to a structure that contains all useful information to process the transmission of a TxNsdu.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00269 :</b>		
<b>Name</b>	CanTpTxPaddingActivation {CANTP_PADDING_ACTIVATION}		
<b>Description</b>	Defines if the transmit frame use padding or not. Definition of Enumeration values: CanTpOn The transmit N-PDU uses padding for SF, FC and the last CF. (N-PDU length is always 8 bytes) CanTpOff The transmit N-PDU does not use padding for SF, CF and the last CF. (N-PDU length is dynamic)		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucEnumerationParamDef		
<b>Range</b>	CANTP_OFF		Padding is not used
	CANTP_ON		Padding is used
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00270 :</b>		
<b>Name</b>	CanTpTxTaType {CANTP_TA_TYPE}		
<b>Description</b>	Declares the communication type of this TxNsdu. Enumeration values: CanTpPhysical. Used for 1:1 communication.		



	CanTpFunctional. Used for 1:n communication.	
<b>Multiplicity</b>	1	
<b>Type</b>	EcucEnumerationParamDef	
<b>Range</b>	CANTP_FUNCTIONAL	Functional request type
	CANTP_PHYSICAL	Physical request type
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X   All Variants
	<b>Link time</b>	--
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>	scope: local	

<b>SWS Item</b>	<b>ECUC_CanTp_00261 :</b>		
<b>Name</b>	CanTpTxNSduRef		
<b>Description</b>	Reference to a Pdu in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanTpNAe	0..1	This container is required for each RxNSdu and TxNSdu with AddressingFormat CANTP_MIXED or CANTP_MIXED29BIT.
CanTpNSa	0..1	This container is required for each RxNSdu and TxNSdu with RxTaType CANTP_PHYSICAL and CanTpAddressingFormat CANTP_EXTENDED. When DynIdSupport is enabled, this container is also required for each TxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT. When DynIdSupport is enabled and GenericConnectionSupport is not enabled, this container is also required for each RxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT.
CanTpNTa	0..1	This container is required for each RxNSdu and TxNSdu with AddressingFormat CANTP_EXTENDED. When DynIdSupport is enabled, this container is also required for each RxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT. When DynIdSupport is enabled and GenericConnectionSupport is not enabled, this container is also required for each TxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT.
CanTpRxFcNPdu	0..1	Used for grouping of the ID of a PDU and the Reference to a PDU.
CanTpTxNPdu	1	Used for grouping of the ID of a PDU and the Reference to a PDU.

### 10.2.10 CanTpTxNPdu

<b>SWS Item</b>	<b>ECUC_CanTp_00274 :</b>	
<b>Container Name</b>	CanTpTxNPdu	
<b>Description</b>	Used for grouping of the ID of a PDU and the Reference to a PDU.	
<b>Configuration Parameters</b>		

<b>SWS Item</b>	<b>ECUC_CanTp_00286 :</b>	
<b>Name</b>	CanTpTxNPduConfirmationPduld	
<b>Description</b>	Handle Id to be used by the CanIf to confirm the transmission of the CanTpTxNPdu to the CanIf module.	
<b>Multiplicity</b>	1	

<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00275 :</b>		
<b>Name</b>	CanTpTxNPduRef		
<b>Description</b>	Reference to a Pdu in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

**No Included Containers**

### 10.2.11 CanTpRxFcNPdu

<b>SWS Item</b>	<b>ECUC_CanTp_00271 :</b>		
<b>Container Name</b>	CanTpRxFcNPdu		
<b>Description</b>	Used for grouping of the ID of a PDU and the Reference to a PDU.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00273 :</b>		
<b>Name</b>	CanTpRxFcNPdul {CANTP_RXFC_NPDU_ID}		
<b>Description</b>	N-PDU identifier attached to the FC N-PDU of this TxNsdu identified by CanTpTxNSdul. Each TxNsdu identifier is linked to one Rx FC N-PDU identifier only. However, in the case of extended addressing format, the same FC N-PDU identifier can be used for several N-SDU identifiers. The distinction is made by means of the N_TA value (first data byte of FC frames).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanTp_00272 :</b>		
<b>Name</b>	CanTpRxFcNPduRef		
<b>Description</b>	Reference to a Pdu in the COM-Stack.		
<b>Multiplicity</b>	1		
<b>Type</b>	Reference to [ Pdu ]		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	--	
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>			

**No Included Containers**

### 10.2.12 CanTpNTa

<b>SWS Item</b>	<b>ECUC_CanTp_00139 :</b>		
<b>Container Name</b>	CanTpNTa{N_Ta}		
<b>Description</b>	This container is required for each RxNSdu and TxNSdu with AddressingFormat CANTP_EXTENDED. When DynIdSupport is enabled, this container is also required for each RxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT. When DynIdSupport is enabled and GenericConnectionSupport is not enabled, this container is also required for each TxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00255 :</b>		
<b>Name</b>	CanTpNTa {CANTP_NTA}		
<b>Description</b>	This parameter contains the transport protocol target address value.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.2.13 CanTpNSa

<b>SWS Item</b>	<b>ECUC_CanTp_00253 :</b>		
<b>Container Name</b>	CanTpNSa{N_Sa}		
<b>Description</b>	This container is required for each RxNSdu and TxNSdu with RxTaType CANTP_PHYSICAL and CanTpAddressingFormat CANTP_EXTENDED. When DynIdSupport is enabled, this container is also required for each TxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT. When DynIdSupport is enabled and GenericConnectionSupport is not enabled, this container is also required for each RxNSdu with AddressingFormat CANTP_NORMALFIXED or CANTP_MIXED29BIT.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanTp_00254 :</b>		
<b>Name</b>	CanTpNSa {CANTP_NSA}		
<b>Description</b>	This parameter contains the transport protocol source address value.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcuIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.2.14 CanTpNAe

<b>SWS Item</b>	<b>ECUC_CanTp_00284 :</b>
<b>Container Name</b>	CanTpNAe{N_Ae}
<b>Description</b>	This container is required for each RxNSdu and TxNSdu with AddressingFormat CANTP_MIXED or CANTP_MIXED29BIT.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CanTp_00285 :</b>		
<b>Name</b>	CanTpNAe {CANTP_NAE}		
<b>Description</b>	This parameter contains the transport protocol address extension value.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

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

### 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.

## 11 Not applicable requirements

**[SWS\_CanTp\_00327]** [These requirements are not applicable to this specification.]

(SRS\_BSW\_00344, SRS\_BSW\_00404, SRS\_BSW\_00405, SRS\_BSW\_00170, SRS\_BSW\_00419, BSW382, BSW383, BSW397, BSW398, BSW399, BSW400, SRS\_BSW\_00375, SRS\_BSW\_00416, SRS\_BSW\_00168, SRS\_BSW\_00423, SRS\_BSW\_00427, SRS\_BSW\_00428, SRS\_BSW\_00429, BSW00431, SRS\_BSW\_00432, SRS\_BSW\_00433, BSW00434, SRS\_BSW\_00422, BSW00420, SRS\_BSW\_00417, SRS\_BSW\_00161, SRS\_BSW\_00162, BSW00324, SRS\_BSW\_00415, SRS\_BSW\_00325, SRS\_BSW\_00326, SRS\_BSW\_00342, SRS\_BSW\_00413, SRS\_BSW\_00347, SRS\_BSW\_00307, SRS\_BSW\_00314, SRS\_BSW\_00361, SRS\_BSW\_00328, SRS\_BSW\_00378, SRS\_BSW\_00172, SRS\_BSW\_00010, SRS\_BSW\_00321, SRS\_BSW\_00341, SRS\_BSW\_00334)