System Specifications

3

Communication

3

Transport Layer

4

Summary

This document specifies the Transport Layer of the KNX System.

Version 01.02.02 is a KNX Approved Standard.

This document is part of the KNX Specifications v2.1.

Document updates

Version	Date	Modifications
1.0	2001.12.18	Preparation of the Approved Standard.
1.1	2007.08.06	**S13 “Extended Frame Formats” – “Transport Layer Extension”** integration completed.
1.1		**S16 “System Broadcast Services”**
1.1	2008.08.08	**AN002 “Rationalisation of the Transport Layer state machine”** integrated.
1.1	2008.12.22	Preparation of the Approved Standard v1.1.
1.1.01	2009.10.01	Editorial update.
1.1.02	2009.10.08	Editorial update.
1.2.00	2009.11.06	**AN118 “cEMI Transport Layer”** integrated.
01.02.01	2013.10.28	Editorial updates for the publication of KNX Specifications 2.1.
01.02.02	2013.11.29	Editorial updates.

References

[01] Chapter 3/3/2 “Data Link Layer General”
[02] Chapter 3/6/3 “External Message Interface”
[03] Chapter 3/8/3 “KNXnet/IP Device Management”
[04] Part 9/3 “Couplers”

Filename: 03_03_04 Transport Layer v01.02.02 AS.docx
Version: 01.02.02
Status: Approved Standard
Savedate: 2013.11.29
Number of pages: 38

Contents

1 Overview - communication modes 4
1.1 Communication Modes 4
1.2 Point-to-Multipoint, Connectionless (Multicast) Communication Mode 4
1.3 Point-to-all-Points Connectionless (Broadcast) Communication Mode 4
1.4 Point-to-all-Points Connectionless (System Broadcast) communication mode 5
1.5 Point-to-Point, Connectionless Communication Mode 5
1.6 Point-to-Point, Connection-Oriented Communication Mode 5
1.7 Algorithm for the Identifier of communication 5

2 TPDU 6

3 Transport Layer Services 7
3.1 General requirements 7
3.2 T_Data_Group Service 7
3.3 T_Data_Tag_Group service 8
3.4 T_Data_Broadcast Service 9
3.5 T_Data_SystemBroadcast 10
3.6 T_Data_Individual 12
3.7 T_Connect Service 13
3.8 T_Disconnect Service 14
3.9 T_Data_Connected Service 15

4 Parameters of Transport Layer 16

5 State Machine of Connection-Oriented Communication Mode 17
5.1 States 17
5.2 Events 18
5.3 Actions 19
5.4 Transition Table of the Connection Oriented Transport Layer State Machine 21
5.4.1 Style 1 21
5.4.2 Style 2 22
5.4.3 Style 3 24
5.4.4 Style 1 Rationalised 25
5.5 State Diagrams 28
5.5.1 Connect and Disconnect 28
5.5.2 Reception of Data 31
5.5.3 Transmission of Data 33

6 Externally Accessible Transport Layer Interface 36
6.1 Transport Layer access via EMI1 and EMI2 36
6.2 cEMI Transport Layer and flow control 36
6.2.1 cEMI Transport Layer 36
6.2.2 Services for controlling the Transport Layer connection 38

1 Overview - communication modes

1.1 Communication Modes

The Transport Layer (Layer-4) shall provide data transmission over different communication modes. These modes shall connect Transport Layer users with each other. The Transport Layer shall provide five different communication modes:

1. point-to-multipoint, connectionless (multicast)
2. point-to-*domain*, connectionless (broadcast)
3. point-to-all-points, connectionless (system broadcast)
4. point-to-point, connectionless
5. point-to-point, connection-oriented

Every communication mode shall provide specific Transport Layer Service Access Points (TSAPs) accessible via different Transport Layer services. Each of these services shall consists of three service primitives, the request(req), the confirm (con) and the indication (ind).

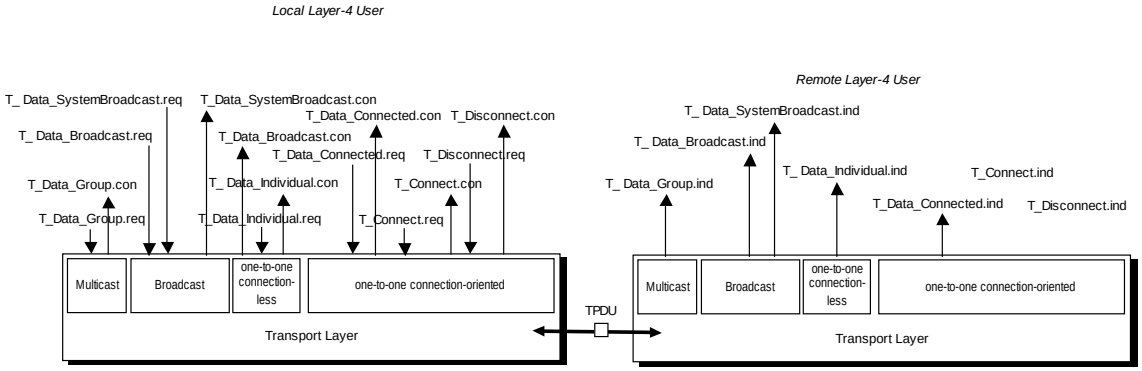


Figure 1 - Interactivity of the Transport Layer

1.2 Point-to-Multipoint, Connectionless (Multicast) Communication Mode

The multicast communication mode shall allow communication between devices belonging to the same group. A device can be member of multiple groups. Any member of the group may be the initiator of a communication. The group shall be identified by its Group Address. The multicast communication mode on Transport Layer shall map TSAPs (group index) to Group Addresses and vice versa.

In Point-to-Multipoint Connectionless (Multicast) Communication Mode the following Transport Layer services shall be available:

T_Data_Group

1.3 Point-to-all-Points Connectionless (Broadcast) Communication Mode

The broadcast communication mode shall connect a single device with all other devices. The single device shall always be transmitter; all other devices shall always be receiver.

The following Transport Layer services shall be available on the Point-to-all-Points Connectionless (Broadcast) Communication Mode:

- T_Data_Broadcast

1.4 Point-to-all-Points Connectionless (System Broadcast) communication mode

The System Broadcast communication mode shall connect a single device with all communication partners. The single end device shall always be a transmitter, the communication partners shall always be receivers.

The following Transport Layer services shall be available on the Point-to-All System Broadcast communication mode:

- T_Data_SystemBroadcast

1.5 Point-to-Point, Connectionless Communication Mode

The Point-to-Point connectionless communication mode shall enable communication between any two single devices.

The following Transport Layer services shall be available on the Point-to-Point, Connectionless Communication Mode:

- T_Data_Individual

1.6 Point-to-Point, Connection-Oriented Communication Mode

The Point-to-Point Connection-Oriented Communication Mode shall allow a reliable communication between any two single devices, within a connection.

The following Transport Layer services shall be available on the Point-to-Point, Connection-Oriented Communication Mode:

- T_Connect
- T_Data_Connected
- T_Disconnect

The user of this communication mode type shall establish the connection before it can be used. The user may release the connection if it is no longer needed. The Transport Layer shall provide a supervision of the connection with a connection-time-out-timer. If the timer expires or if an unrecoverable error occurs, the Transport Layer shall release the connection immediately. Transport Layer shall also provide a reliable end-to-end transmission over Bridges and Routers on the connection-oriented communication mode. T_Data_Connected services shall be repeated up to three times if the T_Data_Connected.req is not acknowledged from the remote Transport Layer entity within an acknowledgment-time-out-time. Repetitions of T_Data_Connected services shall be detected using a sequence number. Parallel services shall not be allowed on a connection-oriented communication mode. The connection-oriented communication mode shall be processed according to the Transport Layer state machine described in clause 5 “State Machine of Connection-Oriented Communication Mode” below.

1.7 Algorithm for the Identifier of communication

The TSAP used in the T_Data_Group service shall be a reference to a Group Address, see [01].

The TSAP used in the T_Data_Individual-service shall be the Individual Address of the communication partner.

The TSAP used in connection-oriented transport communication shall be the identifier of the connection address. The number of possible connections at a time depends on the available internal resources.

2 TPDU

The TPDU is shown in the following Figure 2.

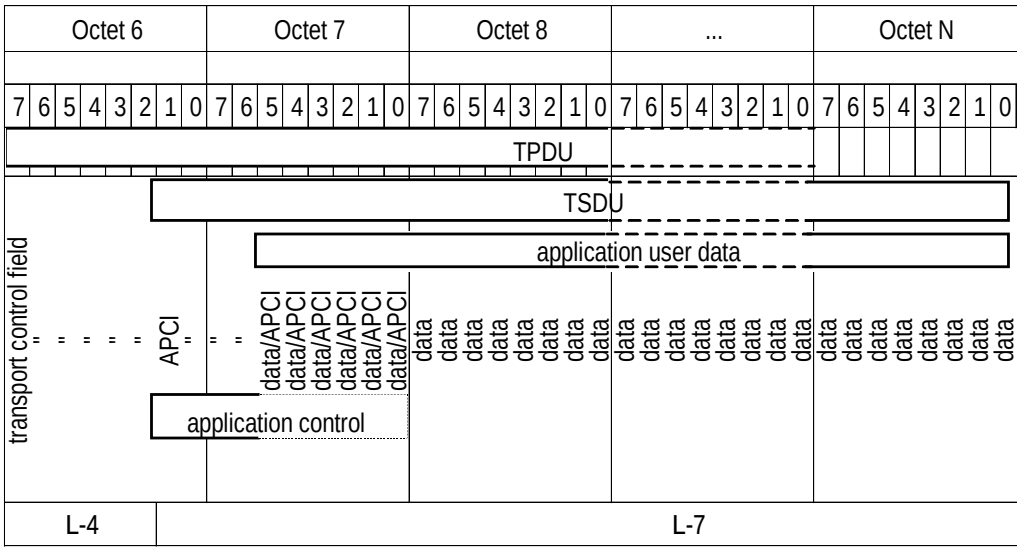


Figure 2 - Format of the TPDU

The TPDU shall correspond to the NPDU, but reduced by the Network Control Field. The Transport Control Field shall be encoded and decoded by Transport Layer and shall contain the Transport Layer service codes and the sequence_number:

Octet 5								Octet 6								
								transport ctrl field								
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	
Address Type (AT)								Data/Control Flag	Numbered							
1								0	0	0	0	0	0			T_Data_Broadcast-PDU (destination_address = 0)
1								0	0	0	0	0	0			T_Data_Group-PDU (destination_address <> 0)
1								0	0	0	0	0	1			T_Data_Tag_Group-PDU
0								0	0	0	0	0	0			T_Data_Individual-PDU
0								0	1	SeqNo	SeqNo	SeqNo	SeqNo			T_Data_Connected-PDU
0								1	0	0	0	0	0	0	0	T_Connect-PDU
0								1	0	0	0	0	0	0	1	T_Disconnect-PDU
0								1	1	SeqNo	SeqNo	SeqNo	SeqNo	1	0	T_ACK-PDU
0								1	1	SeqNo	SeqNo	SeqNo	SeqNo	1	1	T_NAK-PDU

Figure 3 - Transport Control Field

NOTE 1 The encoding BFh is reserved and shall not be used for future extensions.

3 Transport Layer Services

3.1 General requirements

All the Transport Layer services shall provide a confirmation to the requester (user of Transport Layer). The confirmation of the T_Data_Connected service shall indicate that the remote Transport Layer entity did acknowledge the T_Data_Connected.req. The confirmation of the other Transport Layer services shall be caused by the local confirmation of Network Layer.

The user of Transport Layer shall not request a service primitive before the preceding request is confirmed by Transport Layer, i.e. no parallel services are allowed.

3.2 T_Data_Group Service

The T_Data_Group service shall be applied by the user of Transport Layer, to transmit a TSDU over a multicast communication mode to one or more remote partners. The T_Data_Group service shall neither be acknowledged nor be confirmed by the remote Transport Layer entity. The confirmation shall be a local confirmation caused by the N_Data_Group.con of the Network Layer.

The local user of Transport Layer shall prepare a TSDU for the remote user. The local user of Transport Layer shall apply the T_Data_Group.req primitive to pass the TSDU to the local Transport Layer. The destination shall be defined by the TSAP (TSAP).

The local Transport Layer shall accept the service request, shall map the TSAP to the destination Group Address, shall map the arguments to the corresponding arguments of the N_Data_Group.req primitive, shall encode the TSDU to the NSDU and shall pass it with a N_Data_Group.req to the local Network Layer.

The remote Transport Layer shall map a N_Data_Group.ind primitive with NSDU = T_Data_Group-PDU to a T_Data_Group.ind primitive. The remote Transport Layer shall map the destination_address to the TSAP. The argument NSDU shall be mapped to the argument TSDU, the argument priority shall be mapped to the corresponding argument priority of the T_Data_Group.ind primitive.

Prior to passing a T_Data_Group.con primitive to the local user, the local Transport Layer shall need a N_Data_Group.con from the local Network Layer. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Data_Group.con (t_status = ok) to the local user. If the confirmation is negative (n_status = not_ok), the local Transport Layer shall pass a T_Data_Group.con (t_status = not_ok) to the local user indicating that the transmission of the associated N_Data_Group.req did not succeed.

T_Data_Group.req(ack_request, hop_count_type, octet_count, priority, TSAP, tsdu)

ack_request:	Data Link Layer Acknowledge requested or don’t care
hop_count_type:	hop count 7 or Network Layer Parameter
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
TSAP:	identifier of the service access point
tsdu:	this is the user data to be transferred by Transport Layer

T_Data_Group.con(ack_request, hop_count_type, octet_count, priority, source_address, TSAP, tsdu, t_status)

ack_request:		Data Link Layer Acknowledge requested or don't care
hop_count_type:		hop count 7 or Network Layer Parameter
octet_count:		length information as described in Data Link Layer
priority:		system, urgent, normal or low priority
source_address:		the Individual Address of the originator of the message
TSAP:		identifier of the service access point
tsdu:		this is the user data that has been transferred by Transport Layer
t_status:	ok:	T_Data_Group sent successfully with N_Data_Group service
	not_ok:	transmission of the associated N_Data_Group request frame didn't succeed

T_Data_Group.ind(hop_count_type, octet_count, priority, source_address, TSAP, tsdu)

hop_count_type:	hop count equals 7 or not
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
source_address:	the Individual Address of the originator of the message
TSAP:	identifier of the service access point
tsdu:	this is the user data that has been transferred by Transport Layer

3.3 T_Data_Tag_Group service

The T_Data_Tag_Group-service shall be applied by the user of Transport Layer, to transmit a TSDU over a multicast communication mode to one or more remote partners. The T_Data_Tag_Group-service shall neither be acknowledged nor be confirmed by the remote Transport Layer entity. The confirmation shall be a local confirmation caused by the N_Data_Group.con of the Network Layer.

The local user of Transport Layer shall prepare a TSDU for the remote user. The local user of Transport Layer shall apply the T_Data_Tag_Group.req primitive to pass the TSDU to the local Transport Layer. The destination shall be defined by the Destination Address and the Frame Format Parameter.

The local Transport Layer shall accept the service request, shall map the arguments to the corresponding arguments of the N_Data_Group.req primitive, shall encode the TSDU to the NSDU and shall pass it with a N_Data_Group.req to the local Network Layer.

The remote Transport Layer shall map a N_Data_Group.ind primitive with NSDU = T_Data_Tag_Group-PDU to a T_Data_Tag_Group.ind primitive. The argument NSDU shall be mapped to the argument TSDU, the argument priority shall be mapped to the corresponding argument priority of the T_Data_Tag_Group.ind primitive.

Prior to passing a T_Data_Tag_Group.con primitive to the local user, the local Transport Layer shall require a N_Data_Group.con from the local Network Layer. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Data_Tag_Group.con (t_status = ok) to the local user. If the confirmation is negative (n_status = not_ok), the local Transport Layer shall pass a T_Data_Tag_Group.con (t_status = not_ok) to the local user indicating that the transmission of the associated N_Data_Group.req did not succeed.

T_Data_Tag_Group.req(ack_request, destination_address, frame_format, hop_count_type, octet_count, priority, tsdu)

ack_request	This parameter shall be used to indicate whether a Layer-2 acknowledge is mandatory or optional.
destination_address:	This parameter shall be used to contain the Destination Address of the receiver.
frame_format:	This parameter shall be used to indicate the frame format that shall be used to handle the service.
hop_count_type:	hop count 7 or Network Layer Parameter
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
tsdu:	this is the user data to be transferred by Transport Layer

T_Data_Tag_Group.con(hop_count_type, octet_count, priority, source_address, destination_address, frame_format, tsdu, t_status)

hop_count_type:	hop count 7 or Network Layer Parameter	
octet_count:	length information as described in Data Link Layer	
priority:	system, urgent, normal or low priority	
source_address:	the Individual Address of the originator of the message	
destination_address:	Destination Address of the receiver	
Frame Format:	Used Frame Format.	
tsdu:	this is the user data that has been transferred by Transport Layer	
t_status:	ok:	T_Data_Tag_Group sent successfully with N_Data_Group service
	not_ok:	transmission of the associated N_Data_Group request frame didn't succeed

T_Data_Tag_Group.ind(hop_count_type, octet_count, priority, source_address, Destination Address, Frame Format, tsdu)

hop_count_type:	hop count equals 7 or not
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
source_address:	the Individual Address of the originator of the message
Destination Address:	Destination Address of the receiver
Frame Format:	Used Frame Format.
tsdu:	this is the user data that has been transferred by Transport Layer

3.4 T_Data_Broadcast Service

The T_Data_Broadcast service shall be applied by the user of Transport Layer, to transmit a TSDU over a connectionless communication mode to all remote partners. The T_Data_Broadcast service shall neither be acknowledged nor confirmed by any remote Transport Layer entity. The confirmation shall be a local confirmation caused by the N_Data_Broadcast.con of the Network Layer.

The local user of Transport Layer shall prepare a TSDU for all the remote users of Transport Layer. The local user of Transport Layer shall apply the T_Data_Broadcast.req primitive to pass the TSDU to the local Transport Layer. The local Transport Layer shall accept the service request and shall pass it with a N_Data_Broadcast.req to the local Network Layer.

The local Transport Layer shall encode the TSDU to the NSDU and shall map the arguments to the corresponding arguments of the N_Data_Broadcast.req primitive.

The remote Transport Layer shall map a N_Data_Broadcast.ind primitive to a T_Data_Broadcast.ind primitive. The argument NSDU shall be mapped to the argument TSDU, the argument priority shall be mapped to the corresponding argument priority of the T_Data_Broadcast.ind primitive.

Prior to passing a T_Data_Broadcast.con primitive to the local user, the local Transport Layer shall need a N_Data_Broadcast.con from the local Network Layer. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Data_Broadcast.con (t_status = ok) to the local user. If the confirmation is negative (n_status = not_ok), the local Transport Layer shall pass a T_Data_-SystemBroadcast.con (t_status = not_ok) to the local user indicating that the transmission of the associated N_Data_Broadcast.req did not succeed.

T_Data_Broadcast.req	(ack_request, hop_count_type, octet_count, priority, tsdu)	
ack_request:	Data Link Layer Acknowledge requested or don't care	
hop_count_type:	hop count 7 or Network Layer Parameter	
octet_count:	length information as described in Data Link Layer	
priority:	highest, urgent, normal or low priority	
tsdu:	This shall be the user data to be transferred by Transport Layer	

T_Data_Broadcast.con	(ack_request, hop_count_type, octet_count, priority, tsdu, t_status)	
ack_request	Data Link Layer Acknowledge requested or don't care	
hop_count_type	hop count 7 or Network Layer Parameter	
octet_count	length information as described in Data Link Layer	
priority	highest, urgent, normal or low priority	
tsdu	this is the user data that has been transferred by Transport Layer	
t_status	ok:	T_Data_Broadcast sent successfully with N_Data_Broadcast service
	not_ok:	transmission of the associated N_Data_Broadcast request frame did not succeed

T_Data_Broadcast.ind	(hop_count_type, octet_count, priority, source_address, tsdu)	
hop_count_type	hop count equals 7 or not	
octet_count	length information as described in Data Link Layer	
priority	highest, urgent, normal or low priority	
source_address	Individual Address of the device that requested the T_Data_Broadcast service	
tsdu	This shall be the user data that has been transferred by Transport Layer.	

3.5 T_Data_SystemBroadcast

The T_Data_SystemBroadcast service shall be applied by the user of Transport Layer, to transmit a TSDU over a connectionless communication mode to all remote partners. The T_Data_SystemBroadcast service shall neither be acknowledged nor be confirmed by any remote Transport Layer entity. The confirmation shall be a local confirmation caused by the L_SystemBroadcast.con of the Layer-2.

The local user of Transport Layer shall prepare a TSDU for all the remote users of Transport Layer. The local user of Transport Layer shall apply the T_Data_SystemBroadcast.req primitive to pass the TSDU to the local Transport Layer. The local Transport Layer shall accept the service request and pass it with a N_Data_SystemBroadcast.req to the local Network Layer.

The local Transport Layer shall encode the TSDU to the NSDU and map the arguments to the corresponding arguments of the N_Data_SystemBroadcast.req primitive.

The remote Transport Layer shall map a N_Data_SystemBroadcast.ind primitive to a T_Data_System-Broadcast.ind primitive. The argument NSDU shall be mapped to the argument TSDU, the argument priority shall be mapped to the corresponding argument priority of the T_Data_SystemBroadcast.ind primitive.

If the local Transport Layer receives a N_Data_SystemBroadcast.con from the local Network Layer, it shall pass a T_Data_SystemBroadcast.con to the local Transport Layer user. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Data_SystemBroadcast.con (t_status = ok) to the local user. If the confirmation is negative (n_status = not_ok), the local Transport Layer shall pass a T_Data_Broadcast.con (t_status = not_ok) to the local user indicating that the transmission of the associated N_Data_SystemBroadcast.req did not succeed.

T_Data_SystemBroadcast.req (ack_request, hop_count_type, octet_count, priority, tsdu)

- ack_request: This parameter shall be used to indicate whether a Layer-2 acknowledge is mandatory or optional.
- hop_count_type: This parameter shall be used to indicate whether the hop_count shall be set to 7 or if the Network Layer parameter shall be used.
- octet_count: This parameter shall be used to indicate the length information of the requested frame.
- priority: This parameter shall be used to indicate the priority that shall be used to the transmit the requested frame; it shall be “system”, “urgent”, “normal” or “low”.
- tsdu: This parameter shall be used to contain the user data to be transferred by the Transport Layer.

T_Data_SystemBroadcast.con (hop_count_type, octet_count, priority, tsdu, t_status)

- hop_count_type: This parameter shall be used to indicate whether the hop_count of the transmitted frame is set to 7 or if the Network Layer parameter is used.
- octet_count: This parameter shall be used to indicate the length information of the transmitted frame.
- priority: This parameter shall be used to indicate the priority that is used to the transmit the requested frame; it shall be “system”, “urgent”, “normal” or “low”
- tsdu: This parameter shall contain the user data that is transferred by Transport Layer.
- t_status:
  - ok: This value of this parameter shall be used to indicate that the transmission of the T_Data_SystemBroadcast.req is successful.
  - not_ok: This value of this parameter shall be used to indicate that the transmission of the T_Data_SystemBroadcast.req did not succeed.

T_Data_SystemBroadcast.ind (hop_count_type, octet_count, priority, source_address, tsdu)

- hop_count_type: This parameter shall be used to indicate whether the hop count of the received frame equals 7 or not.
- octet_count: This parameter shall be used to contain the length information of the received frame.
- priority: This parameter shall be used to indicate the priority of the received frame; it shall be “system”, “urgent”, “normal” or “low”.
- source_address: This parameter shall be used to indicate the Source Address of the received frame; it shall be the Individual Address of the device that has transmitted the T_Data_SystemBroadcast service.
- tsdu: This parameter shall contain the user data that has been transferred by Transport Layer.

3.6 T_Data_Individual

The T_Data_Individual service shall be applied by the user of Transport Layer, to transmit a TSDU over a connectionless point-to-point communication mode to exactly one remote partner. The T_Data_-Individual service shall neither be acknowledged nor confirmed by the remote Transport Layer entity. The confirmation shall be a local confirmation caused by the N_Data_Individual.con of the Network Layer.

The local user of Transport Layer shall prepare a TSDU for the remote user. The local user of Transport Layer shall apply the T_Data_Individual.req primitive to pass the TSDU to the local Transport Layer. The destination shall be defined by the TSAP. The local Transport Layer shall map the TSAP to the destination Individual Address.

The local Transport Layer shall encode the TSDU to the NSDU and shall map the arguments to the corresponding arguments of the N_Data_Individual.req primitive.

The remote Transport Layer shall map an N_Data_Individual.ind primitive containing a T_DATA_-INDIVIDUAL_REQ_PDU to a T_Data_Individual.ind primitive. The remote Transport Layer shall map the source_address to the TSAP (Individual Address). The argument NSDU shall be mapped to the argument TSDU, the other arguments shall be mapped to the corresponding arguments of the T_Data_Individual.ind primitive.

Prior to passing a T_Data_Individual.con primitive to the local user, the local Transport Layer shall need an N_Data_Individual.con from the local Network Layer. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Data_Individual.con (t_status = ok) to the local user. If the confirmation is negative (n_status = not_ok), the local Transport Layer shall pass a T_Data_-Individual.con (t_status = not_ok) to the local user indicating that the transmission of the associated N_Data_Individual.req did not succeed.

T_Data_Individual.req (ack_request, hop_count_type, octet_count, priority, TSAP, tsdu)

ack_request:	Data Link Layer Acknowledge requested or don't care
hop_count_type:	hop count 7 or Network Layer Parameter
octet_count:	length information as described in Data Link Layer
priority:	highest, urgent, normal or low priority
TSAP:	identifier of the service access point (may be direct the Individual Address of the remote partner)
tsdu:	is the user data to be transferred by Transport Layer

T_Data_Individual.con (ack_request, hop_count_type, octet_count, priority, TSAP, tsdu, t_status)

ack_request:	Data Link Layer Acknowledge requested or don't care	
hop_count_type:	hop count 7 or Network Layer Parameter	
octet_count:	length information as described in Data Link Layer	
priority:	highest, urgent, normal or low priority	
TSAP:	identifier of the service access point (may be direct the Individual Address of the remote partner)	
tsdu:	this is the user data that has been transferred by Transport Layer	
t_status:	ok:	T_Data_Individual sent successfully with N_Data_Individual service
	not_ok:	transmission of the associated N_Data_Individual request frame did not succeed

T_Data_Individual.ind (hop_count_type, octet_count, priority, TSAP, tsdu)

hop_count_type:	hop count equals 7 or not
octet_count:	length information as described in Data Link Layer
priority:	highest, urgent, normal or low priority
TSAP:	identifier of the service access point (may be direct the Individual Address of the remote partner)
tsdu:	is the user data that has been transferred by Transport Layer

3.7 T_Connect Service

NOTE 2 The specification for this service as described below only handles the event handling and frame flow under normal conditions. For exact conditions and exception- and error handling, please refer to clause 5 “State Machine of Connection-Oriented Communication Mode” below.

The T_Connect service shall be applied by the user of Transport Layer, to establish a Transport Layer connection on a connection-oriented point-to-point communication mode. The T_Connect primitives shall be mapped to N_Data_Individual primitives and vice versa according to the Transport Layer state machine described in clause 5 “State Machine of Connection-Oriented Communication Mode”. If the available resources allow for building up a new connection, the local Transport Layer shall accept the service request and shall try to send the T_CONNECT_REQ_PDU to the remote Transport Layer with an N_Data_Individual.req. The destination_address shall be an Individual Address, the priority shall be ‘system’; the ack_request shall be set to true; the octet_count shall be set to 6; the hop_count_type shall be set to ‘Network Layer Parameter’.

If the remote Transport Layer receives N_Data_Individual.ind primitive containing a T_CONNECT_-REQ_PDU and allows for building up a new connection, it shall map the source_address to the TSAP. The other parameters of the N_Data_Individual.ind primitive shall be mapped to the corresponding parameters of the T_Connect.ind primitive. In this case the remote Transport Layer entity shall neither acknowledge nor confirm the T_Connect service to the local Transport Layer. The confirmation shall be a local confirmation, caused by the N_Data_Individual.con of Network Layer.

If the remote Transport Layer receives an N_Data_Individual.ind primitive containing a T_CONNECT_-REQ_PDU and does not allow for building up a new connection, the frame shall not be passed to the remote Transport Layer user. Instead, the remote Transport Layer shall send a T_DISCONNECT_-REQ_PDU using an N_Data_Individual.req primitive to the local Transport Layer.

The local Transport Layer shall need an N_Data_Individual.con from the local Network Layer. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Connect.con (t_status = ok) to the local Transport Layer user. If the confirmation is negative, (n_status = not_ok) the local Transport Layer shall pass a T_Disconnect.ind to the local Transport Layer user indicating that the transmission of the N_Data_Individual.req did not succeed.

T_Connect.req (destination_address, priority)

destination_address:	Indivudual Address of the device to which the transport connection shall be established.
priority:	highest, urgent, normal or low priority

T_Connect.con (destination_address, TSAP, t_status)

destination_address:	Individual Address of the device to which the transport connection was requested	
TSAP:	identifier of the service access point (may be direct the Individual Address of the remote partner)	
t_status:	ok:	T_Connect.req sent successfully with N_Data_Individual service
	not_ok:	transmission of the associated N_Data_Individual request frame did not succeed

T_Connect.ind (TSAP)

TSAP:	identifier of the service access point (may be direct the Individual Address of the remote partner)

3.8 T_Disconnect Service

NOTE 3 The specification for this service as described below only handles the event handling and frame flow under normal conditions. For exact conditions and exception- and error handling, please refer to clause 5 “State Machine of Connection-Oriented Communication Mode” below.

The T_Disconnect service shall be applied by the user of Transport Layer, to release a Transport Layer connection on a connection-oriented point-to-point communication mode. The T_Disconnect primitives shall be mapped to N_Data_Individual primitives and vice versa according to the Transport Layer state machine described in clause 5 “State Machine of Connection-Oriented Communication Mode” below. The local Transport Layer shall accept the service request and shall try to send the T_DISCONNECT_-REQ-PDU to the remote Transport Layer with an N_Data_Individual.req. The TSAP shall be mapped to the Destination Address; the priority shall be set to ‘system’; the ack_request shall be set to true; the octet_count shall be set to 6; the hop_count_type shall be set to ‘Network Layer Parameter’.

The remote Transport Layer shall map an N_Data_Individual.ind primitive containing a T_DISCONNECT_REQ-PDU to a T_Disconnect.ind. The argument Source Address shall be mapped to the argument TSAP. The T_Disconnect service shall neither be acknowledged nor confirmed by the remote Transport Layer entity. The confirmation to the local Transport Layer entity shall be caused by the N_Data_Individual.con of the local Network Layer.

Prior to passing a T_Disconnect.con primitive to the local Transport Layer user, the local Transport Layer shall need an N_Data_Individual.con from the local Network Layer. If the confirmation is positive (n_status = ok), the local Transport Layer shall pass a positive T_Disconnect.con (t_status = ok) to the local Transport Layer user. If the confirmation is negative, (n_status = not_ok) the local Transport Layer shall pass a negative T_Disconnect.con (t_status = not_ok) to the local Transport Layer user indicating that the transmission of the N_Data_Individual.req did not succeed.

The T_Disconnect.ind primitive may also be caused by the Transport Layer entity in order to indicate a protocol error.

T_Disconnect.req (priority, TSAP)

priority	highest, urgent, normal or low priority
TSAP	identifier of the service access point to which the connection shall be released

T_Disconnect.con (priority, TSAP, t_status)

priority	highest, urgent, normal or low priority	
TSAP	identifier of the service access point to which the release of the connection was requested	
t_status	ok:	T_Disconnect.req sent successfully with N_Data_Individual service
	not_ok:	The transmission of the associated N_Data_Individual-frame did not succeed

T_Disconnect.ind (TSAP)

TSAP:	Identifier of the service access point to which the connection is released.

3.9 T_Data_Connected Service

NOTE 4 The specification for this service as described below only handles the event handling and frame flow under normal conditions. For exact conditions and exception- and error handling, please refer to clause5 “State Machine of Connection-Oriented Communication Mode” below.

The T_Data_Connected service shall b applied by the user of Transport Layer, to transmit a TSDU over a Transport Layer connection on a connection-oriented communication mode to a remote partner. The T_Data_Connected service shall be acknowledged with a T_ACK_PDU by the remote Transport Layer entity. The T_Data_Connected primitives shall be mapped to N_Data_Individual primitives and vice versa according to the Transport Layer state machine described in clause 5 “State Machine of Connection-Oriented Communication Mode” below.

The local user of Transport Layer shall prepare a TSDU for the remote user. The local user of Transport Layer shall apply the T_Data_Connected.req primitive to pass the TSDU to the local Transport Layer. The local Transport Layer shall accept the service request only if the connection is established (state OPEN_IDLE). The local Transport Layer shall then map the TSAP to the destination Individual Address and shall try to send the TSDU to the remote Transport Layer with an N_Data_Individual.req; the hop_count_type shall be set to ‘Network Layer Parameter’. The local Transport Layer shall pass a T_Data_Connected.con primitive to the local user that shall indicate either a correct data transfer or it shall pass a T_Disconnect.ind primitive to the local user that shall indicate an erroneous data transfer.

The remote Transport Layer shall only accept the N_Data_Individual.ind with the T_DATA_-CONNECTED_REQ_PDU received, if the connection is established, i.e. in the states OPEN_IDLE, OPEN_WAIT. Therefore mutual T_Data_Connected services shall be allowed on a connection-oriented communication mode. If the N_Data_Individual.ind is accepted, the remote Transport Layer shall map the source Individual Address to the TSAP and shall pass the T_DATA_CONNECTED_REQ_PDU to the remote Transport Layer user. The remote Transport Layer shall confirm the reception by sending an N_Data_Individual.req containing a T_ACK-PDU to the local Transport Layer.

Prior to passing the confirmation to the local user, the local Transport Layer shall need an acknowledgment from the remote Transport Layer. If the acknowledgment is a positive acknowledgment (T_ACK_PDU), the local Transport Layer shall pass a T_Data_Connected.con to the local user. If no acknowledgement is received or if the acknowledgement is a negative acknowledgement, the local Transport Layer shall repeat the transmission of the T_DATA_CONNECTED_REQ_PDU up to 3 times with an acknowledgment time-out time of 3 s. If it fails, the local Transport Layer shall pass a T_Disconnect.ind primitive to the local user indicating that the connection is released (state = CLOSED).

T_Data_Connected.req (octet_count, priority, TSAP, tsdu)

octet_count:	length information as described in Data Link Layer
priority:	highest, urgent, normal or low priority
TSAP:	identifier of the service access point to which the frame shall be sent
tsdu:	this is the user data to be transferred by Transport Layer

T_Data_Connected.con(TSAP) transmission successful

TSAP:	identifier of the service access point to which the frame has been sent

T_Data_Connected.ind(octet_count, priority, TSAP, tsdu)

octet_count:	length information as described in Data Link Layer
priority:	highest, urgent, normal or low priority
TSAP:	identifier of the service access point from which the frame is received
tsdu:	this is the user data that has been transferred by Transport Layer

4 Parameters of Transport Layer

The connection number list shall be the only parameter of Transport Layer. The implementation of this parameter, e.g. table or algorithmic is up to the manufacturer. The latter could be achieved by mapping the address of the communication partner (Group Address or Individual Address with address_type flag) to a 13 bit connection_number.

connection number list:	Group Address table maps connection numbers to Destination Addresses and vice versa.
connection timeout:	time interval of 6 s; timeout to breakdown a connection
acknowledgement timeout:	time interval of 3 s; timer to start a repetition if no acknowledgement was received
max_rep_count	3; maximum of T_Connect.req repetitions

5 State Machine of Connection-Oriented Communication Mode

This state machine is designed for only one connection at a time. To use more than one connection at a time several state machines are needed, one for each connection.

The Transport Layer state machine shall process the services T_Connect, T_Disconnect and T_Data_Connected. Other Transport Layer services shall directly be mapped as described for each individual service independent from the actual state of the Transport Layer state machine. Invalid PDUs are ignored.

Local variables of Transport Layer:

connection_address	used to store the actual Individual Address of the partner (for a given TSAP)
SeqNoSend	binary 4 bit value, used to handle the sequence number of numbered data package for sent frames
SeqNoRcv	binary 4 bit value, used to handle the sequence number of numbered data package for received frames
connection_timeout_timer	time interval of 6 s; starts with transition CLOSED→CONNECTING or OPEN_IDLE; stops with transition OPEN_IDLE→CLOSED; restarts if N_Data_Individual.req is applied in the state machine or a correct N_Data_Individual.ind received
acknowledgment_timeout_timer	time interval of 3 s; starts with transition OPEN_IDLE→OPEN_WAIT stops if a correct T_ACK-PDU is received or with transition into →CLOSED
rep_count	used to count the number of T_DATA_CONNECTED_REQ repetitions

The user of Transport Layer always gets a confirmation for a request. When reading the state machine, keep in mind that the user cannot request a second service primitive before getting the confirmation to the preceding primitive, i.e. no parallel services are allowed.

5.1 States

The state machine has the following states:

CLOSED	There is no connection.
OPEN_IDLE	There is a connection open.
OPEN_WAIT	The state machine is waiting for a T_ACK when data has been sent to the remote partner
CONNECTING	Client only. The state is waiting for an IACK after trying to connect to a remote partner.

5.2 Events

Event label	Event description
E00	N_DATA_INDIVIDUAL_ind,T_CONNECT_REQ_PDU (source_address == connection_address)
E01	N_DATA_INDIVIDUAL_ind, T_CONNECT_REQ_PDU (source_address != connection_address)
E02	N_DATA_INDIVIDUAL_ind, T_DISCONNECT_REQ_PDU (source_address == connection_address)
E03	N_DATA_INDIVIDUAL_ind, T_DISCONNECT_REQ_PDU (source_address != connection_address)
E04	N_DATA_INDIVIDUAL_ind, T_DATA_CONNECTED_REQ_PDU (source_address == connection_address) and (SeqNo_of_PDU == SeqNoRcv)
E05	N_DATA_INDIVIDUAL_ind, T_DATA_CONNECTED_REQ_PDU (source_address == connection_address) and (SeqNo_of_PDU == ((SeqNoRcv –1)&Fh))
E06	N_DATA_INDIVIDUAL_ind, T_DATA_CONNECTED_REQ_PDU (source_address == connection_address) and (SeqNo_of_PDU != SeqNoRcv) and (SeqNo_of_PDU !=((SeqNoRcv-1)&Fh))
E07	N_DATA_INDIVIDUAL_ind, T_DATA_CONNECTED_REQ_PDU (source_address != connection_address)
E08	N_DATA_INDIVIDUAL_ind, T_ACK_PDU (source_address == connection_address) and (SeqNo_of_PDU == SeqNoSend)
E09	N_DATA_INDIVIDUAL_ind, T_ACK_PDU (source_address == connection_address) and (SeqNo_of_PDU != SeqNoSend))
E10	N_DATA_INDIVIDUAL_ind, T_ACK_PDU (source_address != connection_address)
E11	N_DATA_INDIVIDUAL_ind, T_NAK_PDU (source_address == connection_address) and (SeqNo_of_PDU != SeqNoSend)
E11b	N_DATA_INDIVIDUAL_ind, T_NAK_PDU (source_address == connection_address)
E12	N_DATA_INDIVIDUAL_ind, T_NAK_PDU (source_address == connection_address)and (SeqNo_of_PDU == SeqNoSend) and (rep_count < max_rep_count)
E13	N_DATA_INDIVIDUAL_ind, T_NAK_PDU (source_address == connection_address) and (SeqNo_of_PDU == SeqNoSend) and (rep_count >= max_rep_count)
E14	N_DATA_INDIVIDUAL_ind, T_NAK_PDU (source_address != connection_address)
E15	T_DATA_CONNECTED_req
E16	CONNECTION_TIME_OUT_ind

Event label	Event description
E17	ACKNOWLEDGE_TIME_OUT_ind (rep_count < max_rep_count)
E18	ACKNOWLEDGE_TIME_OUT_ind (rep_count >= max_rep_count)
E19	N_DATA_INDIVIDUAL_con T_CONNECT_REQ_PDU IAK = OK (CLIENT ONLY)
E20	N_DATA_INDIVIDUAL_con T_CONNECT_REQ_PDU IAK = NOT OK (CLIENT ONLY)
E21	N_DATA_INDIVIDUAL_con T_DISCONNECT_REQ_PDU
E22	N_DATA_INDIVIDUAL_con T_DATA_CONNECTED_REQ_PDU
E23	N_DATA_INDIVIDUAL_con T_ACK_PDU
E24	N_DATA_INDIVIDUAL_con T_NACK_PDU
E25	T_CONNECT_req (CLIENT ONLY)
E26	T_DISCONNECT_req (CLIENT ONLY)
E27	All other, here not mentioned, messages (e.g. not yet defined TPCI)

5.3 Actions

Action label	Action description
A0	Do nothing
A1	Connection_address = source address of received message Send a T_CONNECT_ind to the user. SeqNoSend=0; SeqNoRcv =0; Start connection timeout timer
A2	Send a N_Data_Individual.req with *T_ACK_PDU, priority = SYSTEM, destination = connection_address, sequence = SeqNoRcv* to the Network Layer (remote device). Increment the SeqNoRcv. Send the received buffer as a T_Data_Connected.ind to the user. Restart the connection timeout timer.
A3	Send an N_Data_Individual.req with *T_ACK_PDU, priority = SYSTEM, destination = connection_address, sequence = sequence of received message* to the Network Layer (remote device). Restart the connection timeout timer.
A4	Send an N_Data_Individual.req with *T_NAK_PDU, priority = SYSTEM, destination = connection_address, sequence = sequence of received message* to the Network Layer (remote device). Restart the connection timeout timer.
A5	Send a T_Disconnect.ind to the user. Stop the acknowledge timeout timer. Stop the connection timeout timer.
A6	Send a N_Data_Individual.req with *T_DISCONNECT_REQ_PDU, priority = SYSTEM, destination = connection_address, sequence = 0* to the Network Layer (remote device). Send a T_Disconnect.ind to the user. Stop the acknowledge timeout timer. Stop the connection timeout timer.

Action label	Action description
A7	Store the received T_Data_Connected.req and send as a N_Data_Individual.req with *T_DATA_CONNECTED_REQ_PDU, destination = connection_address, sequence = SeqNoSend* to the Network Layer (remote device). Clear the rep_count. Start the acknowledge timeout timer. Restart the connection timeout timer.
A8	Stop the acknowledge timeout timer. Increment the SeqNoSend. Send the stored buffer as a T_Data_Connected.con with *cleared errorbits,connection number = 0* to the user. Restart the connection timeout timer.
A8b	Stop the acknowledge timeout timer. Increment the SeqNoSend. Restart the connection timeout timer.
A9	Send the stored message as a N_Data_Individual.req to the Network Layer (remote device). Increment the rep_count. Start the acknowledge timeout timer. Restart the connection timeout timer.
A10	Send a N_Data_Individual.req with *T_DISCONNECT_REQ_PDU Priority = SYSTEM, Destination = source (rbuffer), Sequence = 0* back to sender.
A11	**Store event back and handle after next event. Don't change order of** T_Data_Connected.req events
A12	connection_address=address from T_CONNECT_requ send N_Data_Individual.req with T_CONNECT_REQ_PDU SeqNoSend=0; SeqNoRcv =0; Start connection timeout timer
A13	Send a T_Connect.con to the user.
A14	Send a N_Data_Individual.req with *T_DISCONNECT_REQ_PDU*, *priority = SYSTEM, destination = connection_address, sequence = 0* to the Network Layer (remote device). Send a T_Disconnect.con to the user. Stop the acknowledge timeout timer. Stop the connection timeout timer.
A14b	Send a N_Data_Individual.req with *T_DISCONNECT_REQ_PDU*, *priority = SYSTEM, destination = connection_address, sequence = 0* to the Network Layer (remote device). Stop the acknowledge timeout timer. Stop the connection timeout timer.
A15	Send a T_Disconnect.con to the management user Stop the acknowledge timeout timer. Stop the connection timeout timer.

5.4 Transition Table of the Connection Oriented Transport Layer State Machine

5.4.1 Style 1

For clients that do not accept a connection from the bus.

State should not be entered except in case of an internal error.

Event	State		
	CLOSED	**OPEN_IDLE**	**OPEN_WAIT**
E00	OPEN_IDLE A1	CLOSED A6	CLOSED A6
	CLOSED A10	OPEN_IDLE A0	OPEN_WAIT A0
E01	OPEN_IDLE A1	OPEN_IDLE A10	OPEN_WAIT A10
	CLOSED A10	OPEN_IDLE A0	OPEN_WAIT A0
E02	CLOSED A0	CLOSED A5	CLOSED A5
E03	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E04	CLOSED A10	OPEN_IDLE A2	OPEN_WAIT A2
E05	CLOSED A10	OPEN_IDLE A3	OPEN_WAIT A3
E06	CLOSED A10	OPEN_IDLE A4	OPEN_WAIT A4
E07	CLOSED A10	OPEN_IDLE A10	OPEN_WAIT A10
E08	CLOSED A10	CLOSED A6	OPEN_IDLE A8
E09	CLOSED A10	CLOSED A6	CLOSED A6
E10	CLOSED A10	OPEN_IDLE A10	OPEN_WAIT A10
E11	CLOSED A10	CLOSED A6	CLOSED A6
E12	CLOSED A10	CLOSED A6	OPEN_WAIT A9
E13	CLOSED A10	CLOSED A6	CLOSED A6
E14	CLOSED A10	OPEN_IDLE A10	OPEN_WAIT A10

Event	State		
	CLOSED	**OPEN_IDLE**	**OPEN_WAIT**
E15	CLOSED A5	OPEN_WAIT A7	CLOSED A6
E16	CLOSED A0	CLOSED A6	CLOSED A6
E17	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A9
E18	CLOSED A0	OPEN_IDLE A0	CLOSED A6
E19	CLOSED A0	OPEN_IDLE A13	OPEN_WAIT A13
E20	CLOSED A0	CLOSED A5	CLOSED A5
E21	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E22	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E23	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E24	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E25	OPEN_IDLE A12	CLOSED A6	CLOSED A6
E26	CLOSED A15	CLOSED A14	CLOSED A14
E27	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0

5.4.2 Style 2

State should not be entered except in case of an internal error.

Event	State		
	CLOSED	**OPEN_IDLE**	**OPEN_WAIT**
E00	OPEN_IDLE A1	OPEN_IDLE A0	OPEN_IDLE A0
E01	OPEN_IDLE A1	OPEN_IDLE A0	OPEN_WAIT A0
E02	CLOSED A0	CLOSED A5	CLOSED A5
E03	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0

Event	State		
	CLOSED	**OPEN_IDLE**	**OPEN_WAIT**
E04	CLOSED A0	OPEN_IDLE A2	OPEN_WAIT A2
E05	CLOSED A0	OPEN_IDLE A3	OPEN_WAIT A3
E06	CLOSED A0	OPEN_IDLE A4	OPEN_WAIT A4
E07	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E08	CLOSED A0	CLOSED A6	OPEN_IDLE A8b
E09	CLOSED A0	CLOSED A6	OPEN_WAIT A0
E10	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E11	CLOSED A0	CLOSED A6	OPEN_WAIT A0
E12	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A9
E13	CLOSED A0	OPEN_IDLE A0	CLOSED A6
E14	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E15	CLOSED A0	OPEN_WAIT A7	OPEN_WAIT A11
E16	CLOSED A0	CLOSED A6	CLOSED A6
E17	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A9
E18	CLOSED A0	OPEN_IDLE A0	CLOSED A6
E19	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E20	CLOSED A0	CLOSED A5	CLOSED A5
E21	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E22	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E23	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E24	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E25	OPEN_IDLE A12	OPEN_IDLE A0	OPEN_WAIT A0

Event	State		
	CLOSED	OPEN_IDLE	OPEN_WAIT
E26	CLOSED A0	CLOSED A14b	OPEN_WAIT A11
E27	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0

5.4.3 Style 3

State should not be entered except in case of an internal error.

Event	State			
	CLOSED	OPEN_IDLE	OPEN_WAIT	CONNECTING
E00	OPEN_IDLE A1	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0
E01	OPEN_IDLE A1	OPEN_IDLE A10	OPEN_WAIT A10	CONNECTING A10
E02	CLOSED A0	CLOSED A5	CLOSED A5	CLOSED A5
E03	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0
E04	CLOSED A0	OPEN_IDLE A2	OPEN_WAIT A2	CLOSED A6
E05	CLOSED A0	OPEN_IDLE A3	OPEN_WAIT A3	CONNECTING A3
E06	CLOSED A0	OPEN_IDLE A4	OPEN_WAIT A4	CONNECTING A6
E07	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A10
E08	CLOSED A0	OPEN_IDLE A0	OPEN_IDLE A8	CLOSED A6
E09	CLOSED A0	OPEN_IDLE A0	CLOSED A6	CLOSED A6
E10	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A10
E11	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CLOSED A6
E12	CLOSED A0	CLOSED A6	OPEN_WAIT A9	CLOSED A6
E13	CLOSED A0	CLOSED A6	CLOSED A6	CLOSED A6
E14	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A10
E15	CLOSED A0	OPEN_WAIT A7	OPEN_WAIT A11	CONECTING A11

Event	State			
	CLOSED	**OPEN_IDLE**	**OPEN_WAIT**	**CONNECTING**
E16	CLOSED A0	CLOSED A6	CLOSED A6	CLOSED A6
E17	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A9	CONNECTING A0
E18	CLOSED A0	OPEN_IDLE A0	CLOSED A6	CONNECTING A0
E19	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	OPEN_IDLE A13
E20	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CLOSED A5
E21	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0
E22	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0
E23	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0
E24	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0
E25	CONNECTING A12	CLOSED A6	CLOSED A6	CLOSED A6
E26	CLOSED A15	CLOSED A14	CLOSED A14	CLOSED A14
E27	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0	CONNECTING A0

5.4.4 Style 1 Rationalised

5.4.4.1 Introduction

Within the KNX communication stack the connection-oriented part of the Transport Layer is by far the most complex module. In order to minimize the implementation effort and the required Resources and testing effort, the manufacturer may reduce the complexity of the Transport Layer state machine Style 1, named Style 1 Rationalised.

The styles of the Transport Layer connection-oriented state machine are identical in the operation of the TL. This guarantees the full interoperability of devices with TL implementations based on the different styles. The differences between these styles lie in the way in which error conditions are handled, like the reaction of the TL to the loss of a frame. In case a frame is lost, the TL tries to recover this error and seeks to get into sync with the remote partner again. However this is not possible in all cases: in some situations this synchronization effort leads to an exchange of frames and finally to a timeout, before the connection is closed and the client can start over the download.

The Style 1 Rationalised state machine 1 is not recommended for devices basis on open media nor for EDI devices.

A significant part of the complexity of the TL results from this handling of lost frames. On the other hand it is quite obvious that the risk of an entire frame loss is very low due to the use of Data Link Layer repetitions and acknowledgement.

Therefore Style 1 Rationalised differs in the following from Style 1.

If a data frame or T_ACK frame gets lost the TL shall close the connection immediately.

This allows for a quick restart of the download.

This concept has the following consequences to the required resources:

- Only one timer is required with a timeout of 6 seconds (connection timeout, no acknowledge timeout)
- No repetition counter.
- No sending of T_NAK frames.

5.4.4.2 Differences to Style 1

Style 1 Rationalised shall differ from Style 1 in the following.

- event 06 OPEN_IDLE: go to CLOSED, action A6,
- event 06 OPEN_WAIT: go to CLOSED, action A6,
- event 11: no check of the sequence number
- no support of events 12 and 13
- event 15 OPEN_WAIT: remain in OPEN_WAIT, action A11
- no support of events 17 and 18
- no support of event 24

5.4.4.3 Style 1 (rationalised)

For clients that do not accept a connection from the bus.

State should not be entered except in case of an internal error.

	State		
Event	**CLOSED**	**OPEN_IDLE**	**OPEN_WAIT**
E00	OPEN_IDLE A1	CLOSED A6	CLOSED A6
	CLOSED A10	OPEN_IDLE A0	OPEN_WAIT A0
E01	OPEN_IDLE A1	OPEN_IDLE A10	OPEN_WAIT A10
	CLOSED A10	OPEN_IDLE A0	OPEN_WAIT A0
E02	CLOSED A0	CLOSED A5	CLOSED A5
E03	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E04	CLOSED A10	OPEN_IDLE A2	OPEN_WAIT A2

Event	State		
	CLOSED	**OPEN_IDLE**	**OPEN_WAIT**
E05	CLOSED A10	OPEN_IDLE A3	OPEN_WAIT A3
E06	CLOSED A10	CLOSED A6	CLOSED A6
E07	CLOSED A10	OPEN_IDLE A10	OPEN_WAIT A10
E08	CLOSED A10	CLOSED A6	OPEN_IDLE A8
E09	CLOSED A10	CLOSED A6	CLOSED A6
E10	CLOSED A10	OPEN_IDLE A10	OPEN_WAIT A10
E11b	CLOSED A10	CLOSED A6	CLOSED A6
E12	Event does not exist.	Event does not exist.	Event does not exist.
E13	Event does not exist.	Event does not exist.	Event does not exist.
E14	CLOSED A10	OPEN_IDLE A10	OPEN_WAIT A10
E15	CLOSED A5	OPEN_WAIT A7	OPEN_WAIT A11
E16	CLOSED A0	CLOSED A6	CLOSED A6
E17	Event does not exist.	Event does not exist.	Event does not exist.
E18	Event does not exist.	Event does not exist.	Event does not exist.
E19	CLOSED A0	OPEN_IDLE A13	OPEN_WAIT A13
E20	CLOSED A0	CLOSED A5	CLOSED A5
E21	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E22	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E23	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0
E24	Event does not exist.	Event does not exist.	Event does not exist.
E25	OPEN_IDLE A12	CLOSED A6	CLOSED A6
E26	CLOSED A15	CLOSED A14	CLOSED A14

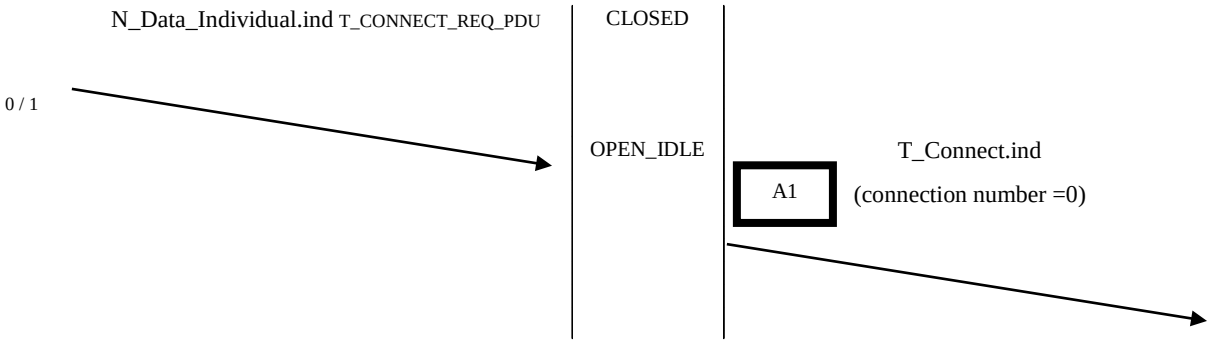
Event	State		
	CLOSED	OPEN_IDLE	OPEN_WAIT
E27	CLOSED A0	OPEN_IDLE A0	OPEN_WAIT A0

5.5 State Diagrams

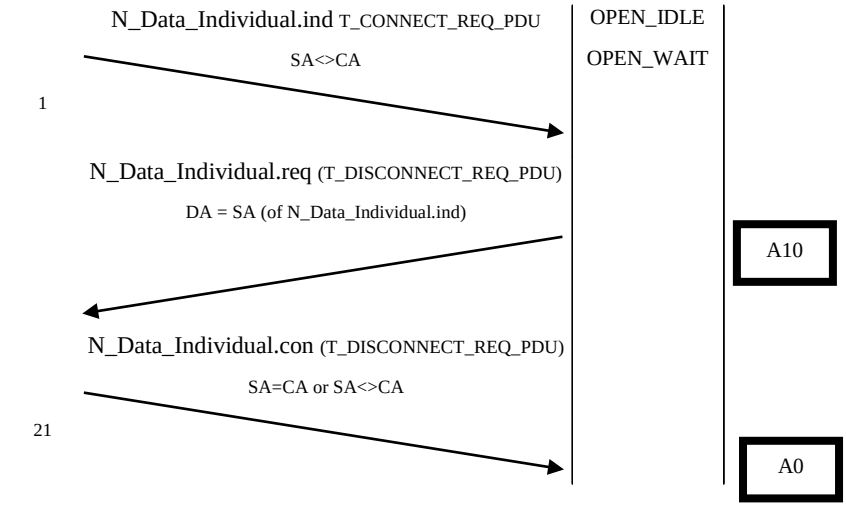
The state diagram below gives examples for a sub-set of possible events. For the actual specification and the specification of the event handling not given below, please refer to paragraph 5 “State Machine of Connection-Oriented Communication Mode” above.

5.5.1 Connect and Disconnect

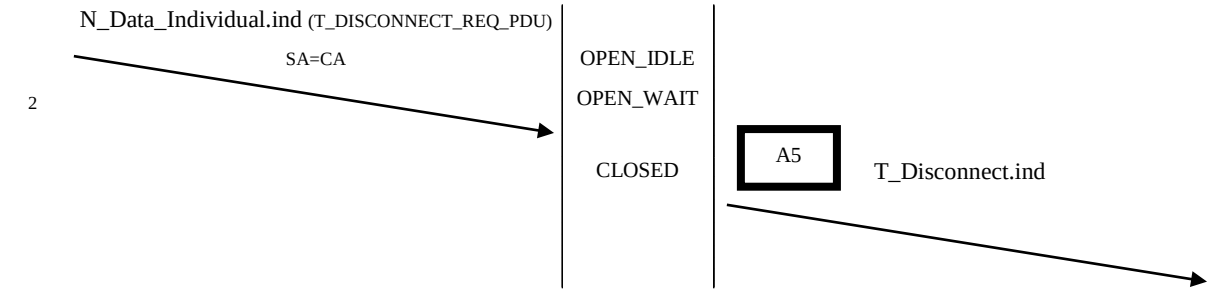
5.5.1.1 Connect from a remote Device



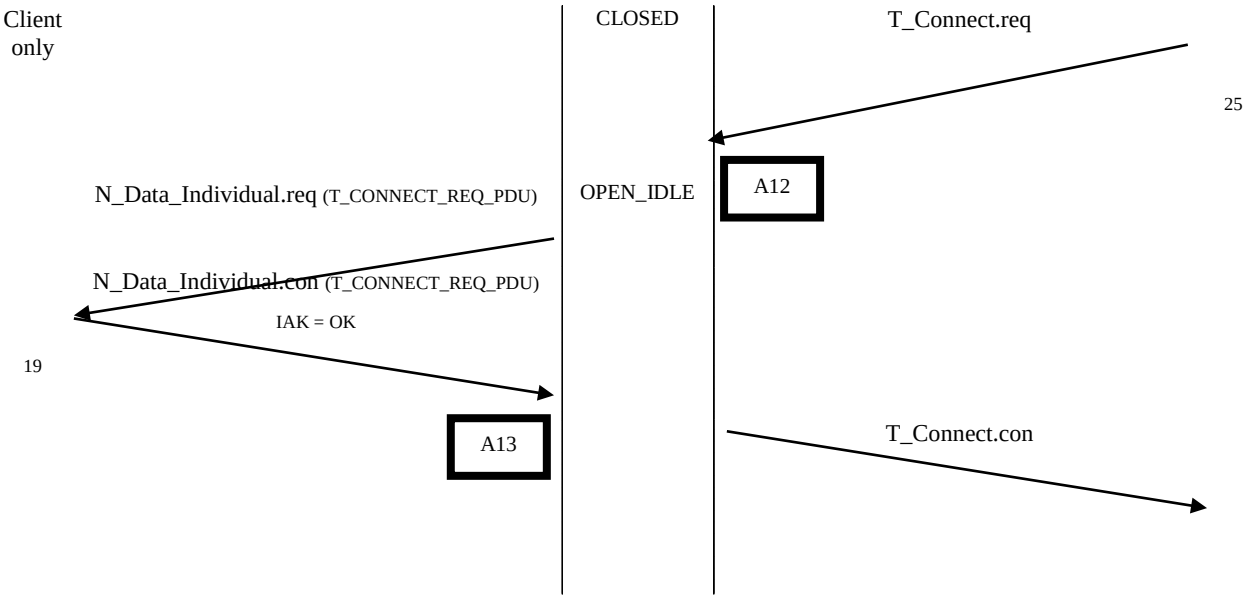
5.5.1.2 Connect from a remote Device during an existing Connection



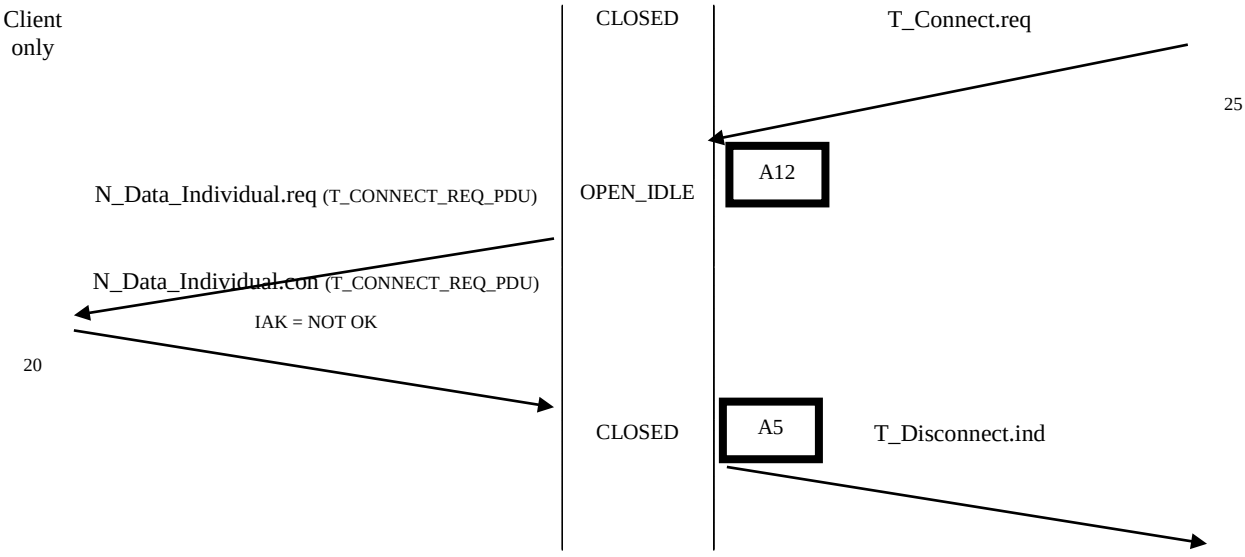
5.5.1.3 Disconnect from a remote Device



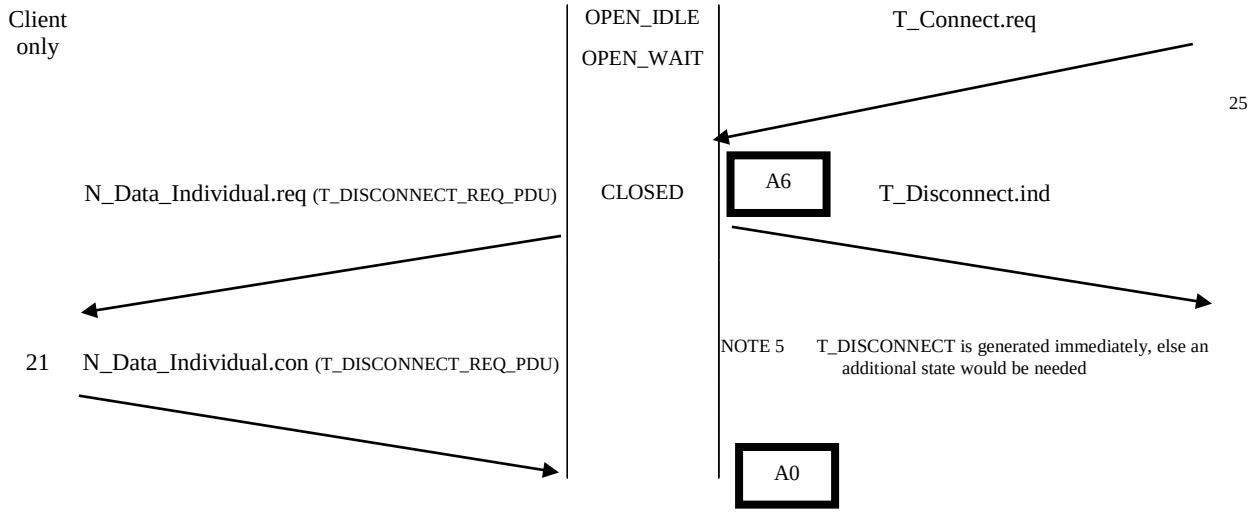
5.5.1.4 Connect from the local User to an existing Device



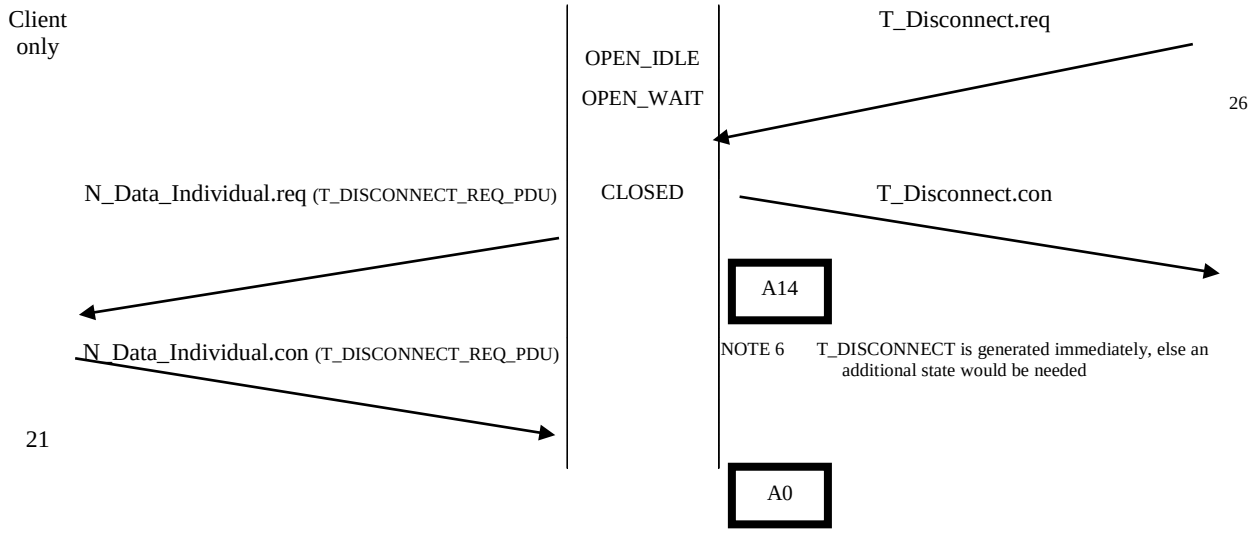
5.5.1.5 Connect from the local User to an non existing Device



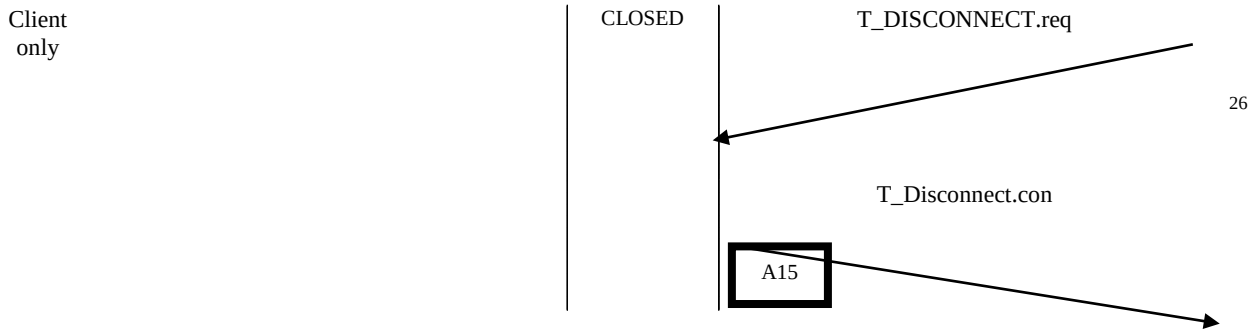
5.5.1.6 Connect from the local User during an existing Connection



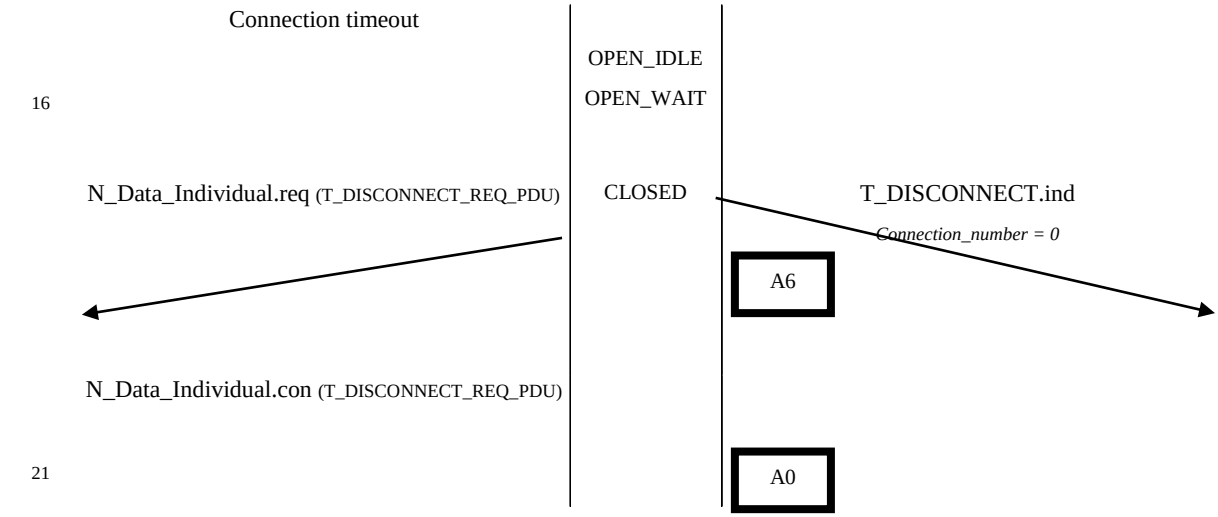
5.5.1.7 Disconnect from the local User



5.5.1.8 Disconnect from the local User without an existing Connection

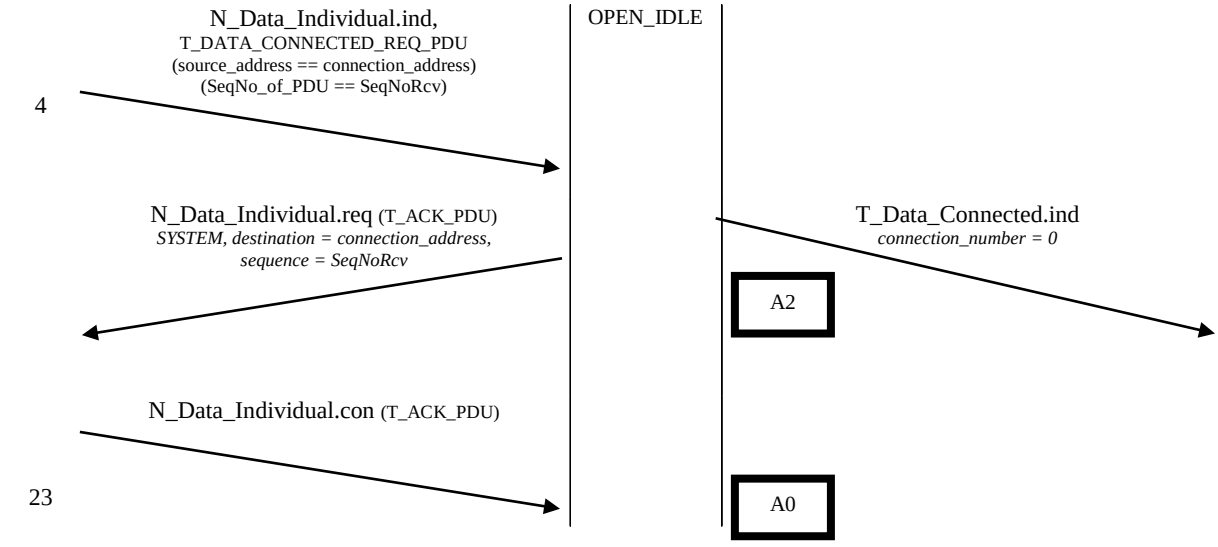


5.5.1.9 Connection timeout

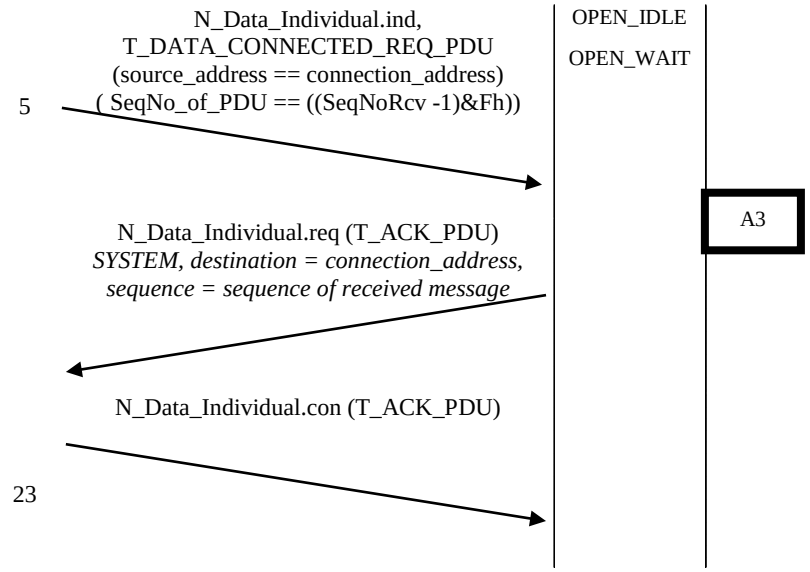


5.5.2 Reception of Data

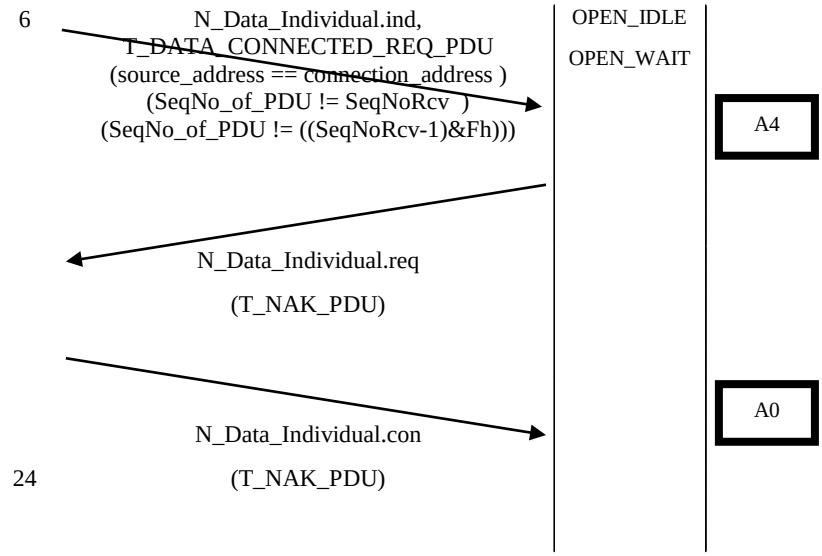
5.5.2.1 Reception of a correct N_Data_Individual



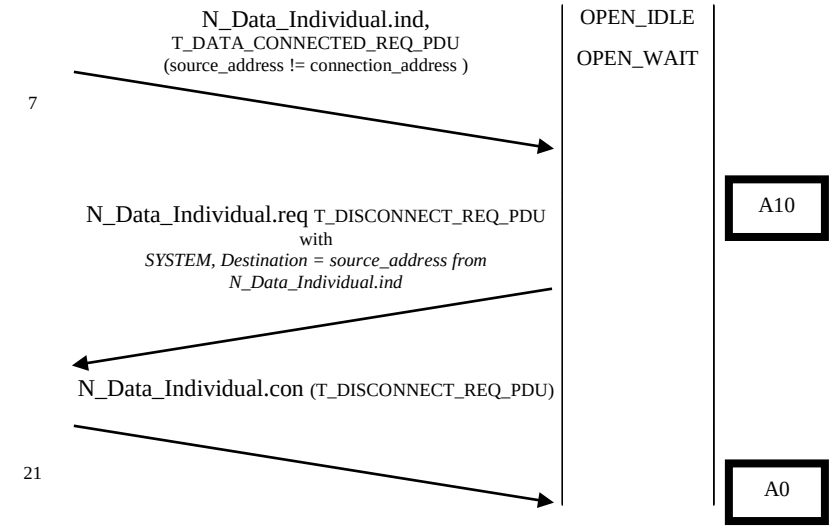
5.5.2.2 Reception of a repeated N_Data_Individual



5.5.2.3 Reception of data N_Data_Individual with wrong sequence Number

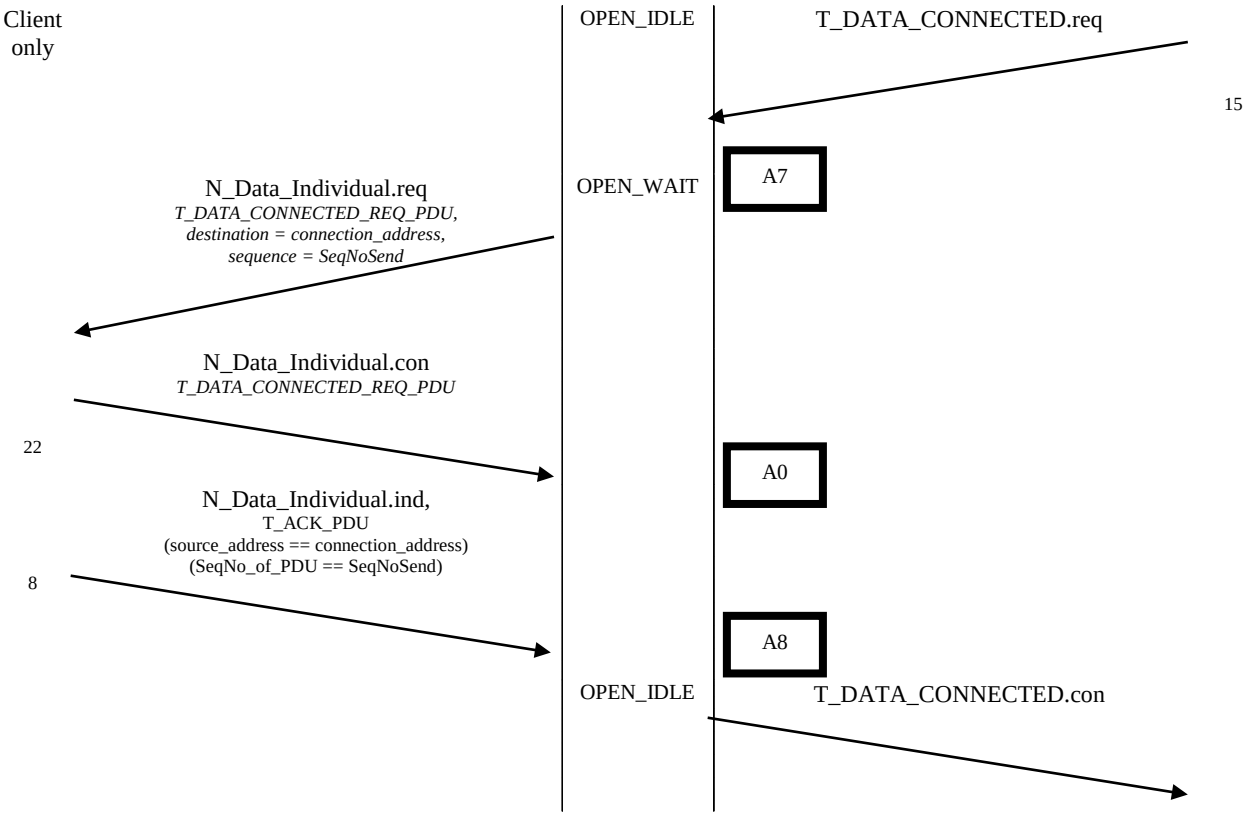


5.5.2.4 Reception of data N_Data_Individual with wrong Source Address

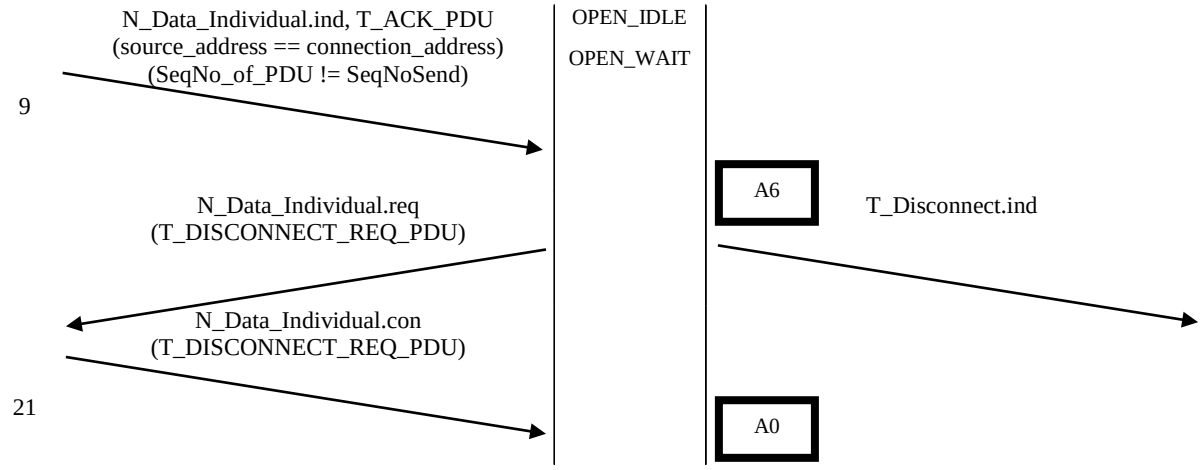


5.5.3 Transmission of Data

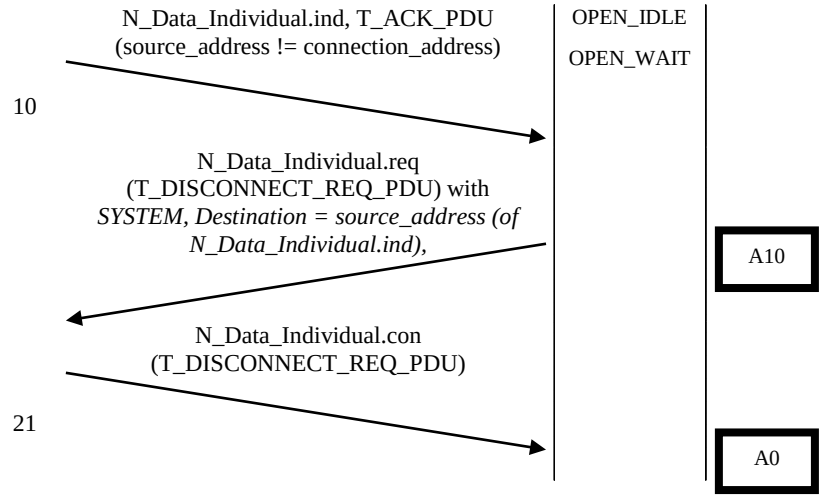
5.5.3.1 T_DATA-Request from the local User



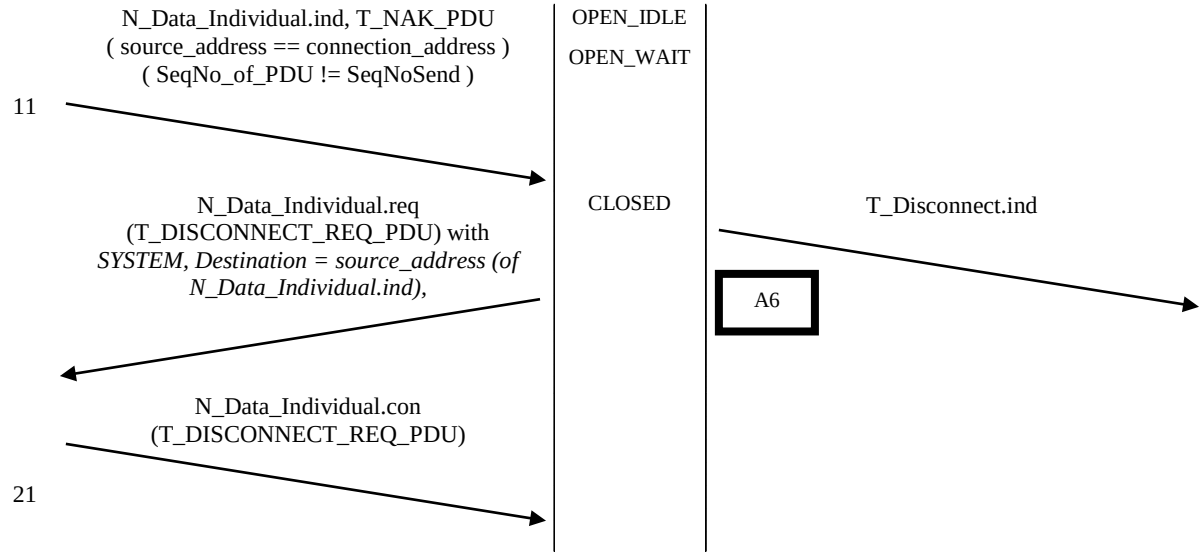
5.5.3.2 Reception of a T_ACK_PDU with wrong Sequence Number



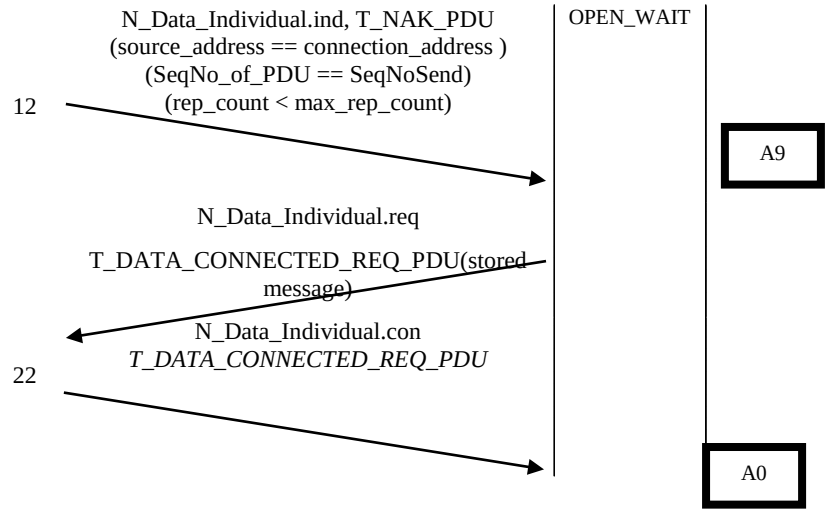
5.5.3.3 Reception of T_ACK_PDU with wrong Connecton Address



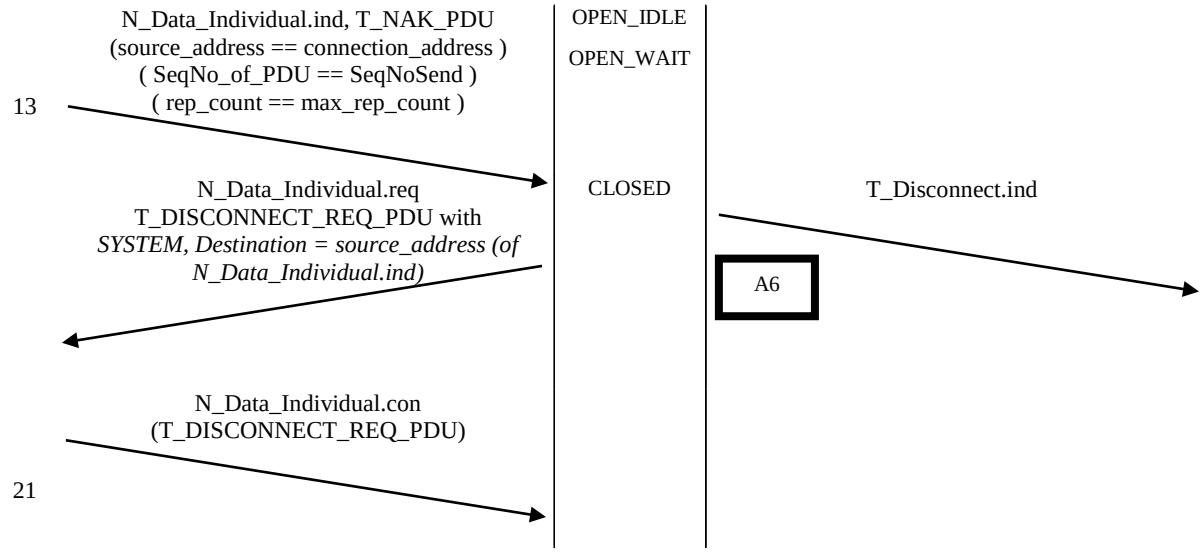
5.5.3.4 Reception of T_NAK_PDU with wrong Sequence Number



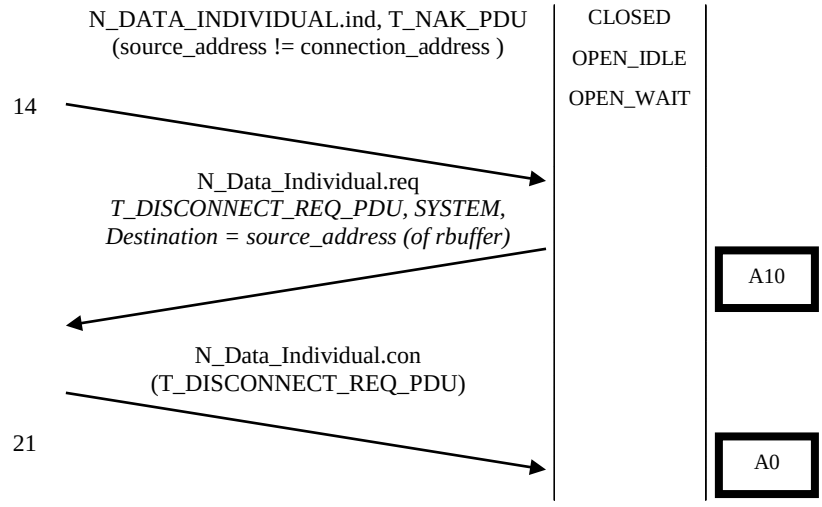
5.5.3.5 Reception of T_NAK_PDU with correct Sequence Number



5.5.3.6 Reception of T_NAK_PDU and maximum Number of Repetitions is reached



5.5.3.7 Reception of T_NAK_PDU with wrong Connection Address



6 Externally Accessible Transport Layer Interface

6.1 Transport Layer access via EMI1 and EMI2

The Transport Layer services can be made available to an external user application by switching off layers 7 and 8 and the internal user application. See [02], paragraph "Transport Layer EMI" for the Transport Layer message formats and paragraph "Layer Access Management" how to switch to the Transport Layer EMI.

6.2 cEMI Transport Layer and flow control

6.2.1 cEMI Transport Layer

6.2.1.1 Goal of the cEMI Transport Layer (informative)

One goal in the definition of local management via cEMI is to keep most of the routines for remote and local device programming the same. To this, it is necessary that the cEMI client can send the same APDUs to the Management in the cEMI server devices as the APDUs that are sent to a further device. This means that download procedures for remote and local programming via cEMI differ only in setting the OSI layer that is accessed in the Management Server. The OSI layer that is selected by the Management Client shall control the message format and message flow.

6.2.1.2 Definition of the cEMI Transport Layer

The cEMI Transport Layer shall provide the Transport Layer interface for the following Transport Layer services; these are further on named “cEMI Transport Layer services”:

1. T_Data_Individual
2. T_Data_Connected

The cEMI Transport Layer shall additionally provide the following Transport Layer service primitives to the Transport Layer user:

1. T_Connect.ind, and
2. T_Disconnect.ind

The cEMI Transport Layer shall be a Transport Layer instance provided by a cEMI Server in the device.

6.2.1.3 cEMI Transport Layer mode

The cEMI Transport Layer Mode shall be an operation mode of the Transport Layer.

cEMI Transport Layer is inactive

- All Transport Layer services shall be handled by the normal Transport Layer.
- cEMI Transport Layer services received by the cEMI Server from the cEMI Client shall by the cEMI Server be ignored and thrown away.

cEMI Transport Layer is active

If the cEMI Transport Layer mode is active this shall denote the following.

a. For the cEMI Transport Layer

1. The cEMI Transport Layer shall be active: it shall handle the cEMI Transport Layer services T_Data_Individual and T_Data_Connected as specified in [02].

b. For the normal Transport Layer

2. All other Transport Layer services shall not be supported by the cEMI Transport Layer. These shall be handled further by the normal Transport Layer in the device.

   NOTE 7 This means that activating the cEMI Transport Layer does not inhibit the sending or receiving of normal group communication (T_Data_Group, T_Data_Tag_Group) neither of broadcast communication (T_Data_Broadcast).

c. For the cEMI Server

3. The cEMI Server shall further support the cEMI frames for local device management without limitation (M_PropRead, M_PropWrite, M_PropInfo, M_Reset).
4. The cEMI Server shall not transmit any other frame to the cEMI Client (e.g. L_Data) and shall ignore any other frame that may be received from the cEMI Client.
5. The cEMI Server shall pass every cEMI Transport Layer frame received from the cEMI Client via the cEMI Transport Layer to the Application Layer in the cEMI device itself. This means that Transport Layer services for remote programming have to be mapped to services for local programming inside the target device.

   EXAMPLE A management client wants to write data to a device using connection oriented Transport Layer. This will lead to a “T_Data_Connected.req” from the cEMI client to the cEMI server. If the cEMI server is switched to cEMI Transport Layer it changes the service from “T_Data_Connected.req” to “T_Data_Connected.ind” and passes it to the user of the cEMI Transport layer.

6. The cEMI Server shall pass every cEMI Transport Layer frame received by the cEMI Transport Layer to the cEMI client.

The cEMI Transport Layer mode can thus be considered as a switch that controls the flow of the cEMI Transport Layer services (T_Data_Individual and T_Data_Connected) as shown in Figure 4. This switch is not modelled any further.

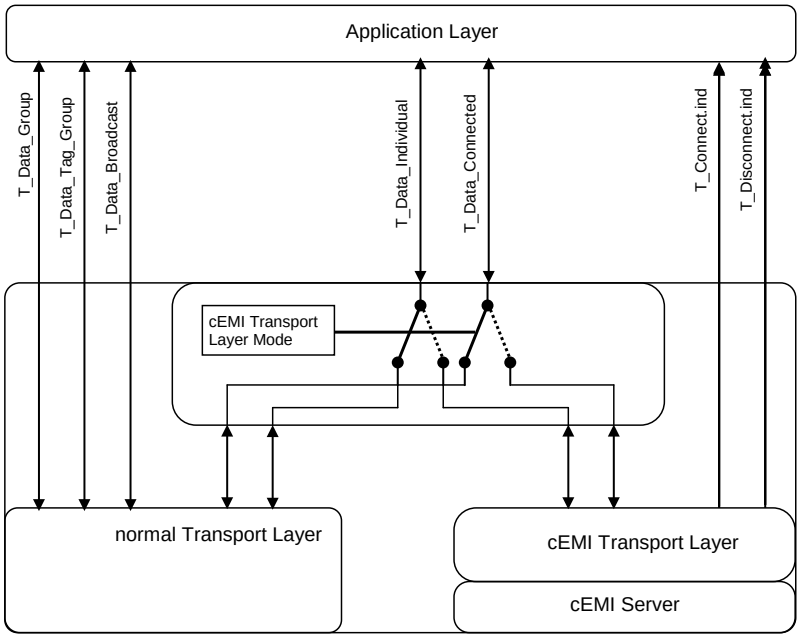


Figure 4 – cEMI Transport Layer Mode

6.2.1.4 Controlling the cEMI Transport Layer

The control of the cEMI Transport Layer mode shall depend on the cEMI host protocol (see [04] for USB and [03] for KNXnet/IP).

6.2.2 Services for controlling the Transport Layer connection

The cEMI Transport Layer does support the T_Data_Connected service to the Transport Layer user but does not provide the Connection Oriented Transport Layer State Machine. It does not support the Transport Layer frames T_Connect-PDU and T_Disconnect-PDU. The communication of the T_Data_Individual- and T_Data_Connected frames is thus not natively governed by cEMI. This requires that the cEMI host protocol (USB, IP…) provides functionality for.

- opening and closing connections, and
- acknowledging of the cEMI frames, if required and if supported by the cEMI host protocol
- supervising connection timeouts on the host protocol
- controlling sequencing of the cEMI messages (if supported by the cEMI host protocol).