International Standard

ISO 15765-2

Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) —

Part 2: Transport protocol and network layer services

Véhicules routiers — Communication de diagnostic sur gestionnaire de réseau de communication (DoCAN) —

Partie 2: Protocole de transport et services de la couche réseau

Fourth edition
2024-04

Reference number
ISO 15765-2:2024(en)

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

ISO draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents. ISO shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 31, *Data communication*.

This fourth edition cancels and replaces the third edition (ISO 15765-2:2016), which has been technically revised.

The main changes are as follows:

— restructured the document to achieve compatibility with OSI 7-layers model;

— introduced T_Data abstract service primitive interface to achieve compatibility with ISO 14229-2;

— moved all transport layer protocol-related information to Clause 9;

— clarification and editorial corrections.

A list of all parts in the ISO 15765 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO 15765 series defines common requirements for vehicle diagnostic systems using the controller area network (CAN), as specified in the ISO 11898 series.

The ISO 15765 series presumes the use of external test equipment for inspection, diagnostics, repair and other possible use cases connected to the vehicle.

This document defines the requirements to enable the in-vehicle CAN network to successfully establish, maintain and terminate communication with the devices externally connected to the diagnostic link connector.

This document has been structured according to the open systems interconnection (OSI) basic reference model, in accordance with ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers. When mapped on this model, the OSI layer 4 and OSI layer 3 framework requirements specified or referenced in the ISO 15765 series are structured according to Figure 1, which shows the related documents of OSI layer 4 and OSI layer 3.

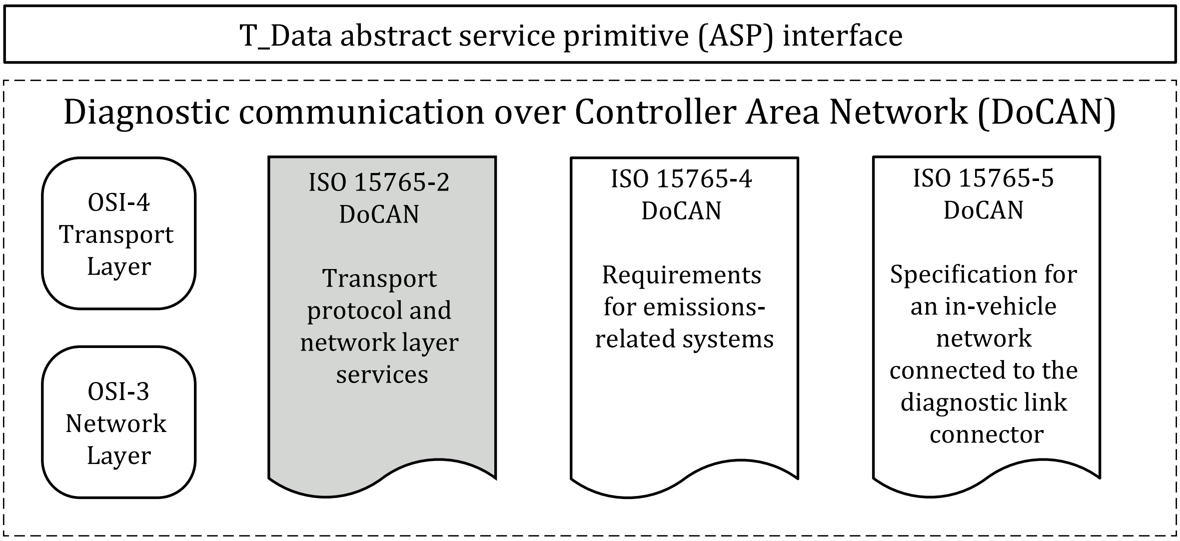


Figure 1 — DoCAN document reference according to the OSI model

Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) —

Part 2: Transport protocol and network layer services

1 Scope

This document specifies a transport and network layer protocol with transport and network layer services tailored to meet the requirements of CAN-based vehicle network systems on controller area networks as specified in ISO 11898-1.

The diagnostic communication over controller area network (DoCAN) protocol supports the standardized abstract service primitive interface as specified in ISO 14229-2 (UDS).

This document supports different application layer protocols such as:

— enhanced vehicle diagnostics (emissions-related system diagnostics beyond legislated functionality, non-emissions-related system diagnostics);

— emissions-related on-board diagnostics (OBD) as specified in the ISO 15031 series and SAE J1979 series;

— world-wide harmonized on-board diagnostics (WWH-OBD) as specified in the ISO 27145 series; and

— end of life activation of on-board pyrotechnic devices (the ISO 26021 series).

The transport protocol specifies an unconfirmed communication.

NOTE This document does not determine whether CAN CC, CAN FD or both are recommended or required to be implemented by other standards referencing this document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 7498-1, *Information technology — Open Systems Interconnection — Basic Reference Model: The Basic Model*

ISO 11898-1[1)], *Road vehicles — Controller area network (CAN) — Part 1: Data link layer and physical signalling*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 7498-1, ISO 11898-1 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

1) Third edition under preparation. Stage at the time of publication: ISO/FDIS 11898-1:—.

3.1
CAN_DL
CAN frame data length
physical length of CAN frame data/*payload* (3.2) in bytes

Note 1 to entry: See Table 2.

3.2
payload
synonym for (CAN) data field as specified in ISO 11898-1

3.3
TX_DL
transmit data link layer data length
parameter configuring the maximum usable *payload* (3.2) length in bytes of the data link layer in the transmitter for the application that implements the network layer

Note 1 to entry: The TX_DL is a fixed configuration value on the sender side for the PDU transmission.

3.4
RX_DL
received data link layer data length
parameter retrieving the maximum usable *payload* (3.2) length in bytes of the data link layer in the receiver for the application that implements the network layer

Note 1 to entry: The RX_DL value is retrieved from the FirstFrame (FF) *CAN_DL* (3.1) of a segmented PDU and is used to verify the correct data length of ConsecutiveFrames (CF).

4 Symbols and abbreviated terms

4.1 Symbols

$C_{\mathrm{TL\_BUFFER\_OVFLW}}$	ComParam transport layer buffer overflow
$C_{\mathrm{TL\_CFSN}}$	ComParam transport layer consecutive frame sequence number
$C_{\mathrm{TL\_DLC}}$	ComParam transport layer data length code
$C_{\mathrm{TL\_ERROR}}$	ComParam transport layer error
$C_{\mathrm{TL\_FCFS}}$	ComParam transport layer flow control flow status
$C_{\mathrm{TL\_FCBS}}$	ComParam transport layer flow control block size
$C_{\mathrm{TL\_FCFS(CTS)}}$	ComParam transport layer flow control flow status continue to send
$C_{\mathrm{TL\_FCFS(OVFLW)}}$	ComParam transport layer flow control flow status overflow
$C_{\mathrm{TL\_FCSTmin}}$	ComParam transport layer flow control separation time minimum
$C_{\mathrm{TL\_FCFS(WAIT)}}$	ComParam transport layer flow control flow status wait
$C_{\mathrm{TL\_INVALID\_FS}}$	ComParam transport layer error invalid flow status
$C_{\mathrm{TL\_OK}}$	ComParam transport layer ok
$C_{\mathrm{TL\_RX\_ON}}$	ComParam transport layer receiver error to indicate that the receiving entity did not accept flow control parameter changes during this segmented message reception
$C_{\mathrm{TL\_TIMEOUT\_A}}$	ComParam transport layer timeout A sender and receiver

$C_{\mathrm{TL\_TIMEOUT\_Bs}}$	ComParam transport layer sender timeout B sender
$C_{\mathrm{TL\_TIMEOUT\_Cr}}$	ComParam transport layer receiver timeout C receiver
$C_{\mathrm{TL\_UNEXP\_PDU}}$	ComParam transport layer error unexpected protocol data unit
$C_{\mathrm{TL\_WFT\_OVRN}}$	ComParam transport layer wait frame transmissions overrun
$C_{\mathrm{TL\_WFTmax}}$	ComParam transport layer flow status wait frame transmissions maximum
$C_{\mathrm{TL\_WRONG\_PARAMETER}}$	ComParam transport layer error wrong parameter
$C_{\mathrm{TL\_WRONG\_SN}}$	ComParam transport layer error wrong segment number
$C_{\mathrm{TL\_WRONG\_VALUE}}$	ComParam transport layer error wrong value
t	time
$t_{\mathrm{TL\_Ar}}$	timing parameter transport layer receiver timing value Ar
$t_{\mathrm{TL\_As}}$	timing parameter transport layer sender timing value As
$t_{\mathrm{TL\_Br}}$	timing parameter transport layer receiver timing value Br
$t_{\mathrm{TL\_Bs}}$	timing parameter transport layer sender timing value Bs
$t_{\mathrm{TL\_Cr}}$	timing parameter transport layer receiver timing value Cr
$t_{\mathrm{TL\_Cs}}$	timing parameter transport layer sender timing value Cs

4.2 Abbreviated terms

For the purposes of this document, the following abbreviated terms apply.

AE	address extension
AI	address information
CAN	controller area network
CAN CC	CAN with static arbitration and data phase bit rate
CAN_DL	CAN frame data length
CAN FD	CAN with flexible data phase bit rate
CF	consecutive frame
ChangeParameter	layer service name
ComParam	communication parameter
CTS	continue to send
Data.	abstract service primitive service name
DoCAN	diagnostic communication over CAN
ECU	electronic control unit
FC	flow control

FF	first frame
FF_DL	first frame data length in bytes
FMI	failure mode indicator
Mtype	message type
N/A	not applicable
PCI	protocol control information
PCItype	protocol control information type
PDU	protocol data unit
SA	source address
SDU	service data unit
TA	target address
TAtype	target address type
NL	network layer
OBD	on-board diagnostics
OSI	Open Systems Interconnection
PCI	protocol control information
RTR	remote transmission request
RX_DL	received data link layer data length
SF	single frame
SF_DL	single frame data length in bytes
SN	sequence number
SPN	suspect parameter number
TX_DL	transmit data link layer data length
UDS	unified diagnostic services
WWH-OBD	world-wide harmonized OBD

5 Conventions

This document is based on the conventions discussed in the OSI service conventions (ISO/IEC 10731) as they apply for diagnostic services.

6 ISO 11898-1 CAN data link layer extension

6.1 CAN CC and CAN FD frame feature comparison

ISO 11898-1 specifies variable length CAN frames with a maximum payload size dependent on the protocol device used. A CAN CC protocol device can transmit/receive frames with payload sizes ranging from 0 byte to 8 byte per frame.

A CAN FD (flexible data rate) protocol device can transmit/receive frames with payload sizes from 0 byte to 64 byte. A CAN FD protocol device is also capable of transmitting/receiving CAN CC frames.

Therefore, the segmented transfer of data using FirstFrame (FF), FlowControl (FC) and ConsecutiveFrame (CF) type frames shall support a variable configurable payload length without changing the original protocol concept.

Table 1 outlines the different features of the CAN frame types provided by ISO 11898-1.

Table 1 — CAN frame feature comparison

RefNo	Feature	CAN CC	CAN FD
#1	Payload length 0 to 8 bytes: data length code (DLC) 0 to 8	Yes	Yes
#2	Payload length 8 bytes: data length code (DLC) 9 to 15[a]	Yes	No
#3	Payload length 12 to 64 bytes[b]: data length code (DLC) 9 to 15	No	Yes
#4	Different bit rates supported for the arbitration and data phases of a CAN frame	No	Yes
#5	Remote transmission request (RTR)	Yes	No

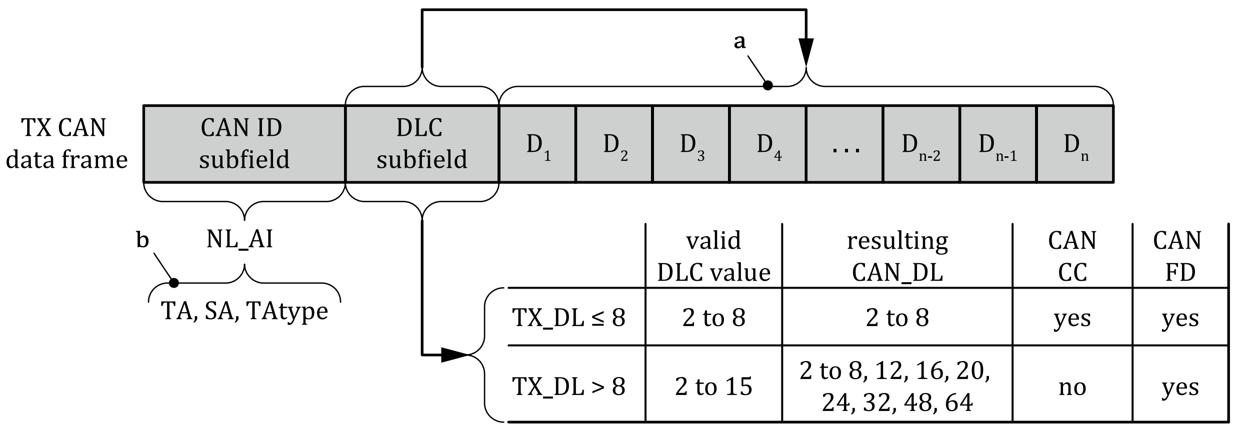
[a] For CAN CC, the DLC values 9 to 15 are automatically reduced to the value of 8 which leads to the maximum possible CAN_DL for CAN CC.

[b] CAN FD does not support all payload lengths between 8 bytes and 64 bytes (e.g. a CAN FD frame with 10 meaningful data bytes requires a payload length of 12 bytes); see Table 2.

6.2 Mapping of transport and network layer attributes to CAN data frames

Figure 2 shows the mapping of CAN parameters onto the data link layer addressing information NL_AI. It illustrates the validity and applicability of transport/network layer parameters and the resulting support of CAN CC versus CAN FD data link layer.

Figure 2 describes this for the example of using either normal or normal fixed addressing. For extended addressing and mixed addressing, the concept in general also applies but the mapping of the NL_AI parameter onto the CAN frame differs.



Key

[a] DLC value results in a CAN_DL value (n), which is the physical length of a CAN frame data/payload; in the receiver, CAN_DL is used to determine the sender TX_DL value.

[b] The shown NL_AI mapping is an example for normal and normal fixed addressing only.

For 11-bit CAN identifiers, the mapping of the NL_AI target address (TA) and source address (SA) into a CAN identifier is implied.

Figure 2 — Illustration of transport and network layer attributes mapping to the CAN data frame subfields

Table 2 shows the data length code value between CAN CC and CAN FD.

Table 2 — CAN CC/CAN FD data length comparison table

Data length code (DLC)	CAN CC data length (CAN_DL)	CAN FD data length (CAN_DL)
0	0	0
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	8[a]	12
10	8[a]	16
11	8[a]	20
12	8[a]	24
13	8[a]	32
14	8[a]	48
15	8[a]	64

[a] For CAN CC, the DLC values 9 to 15 are automatically reduced to the value of 8 which leads to the maximum possible CAN_DL for CAN CC.

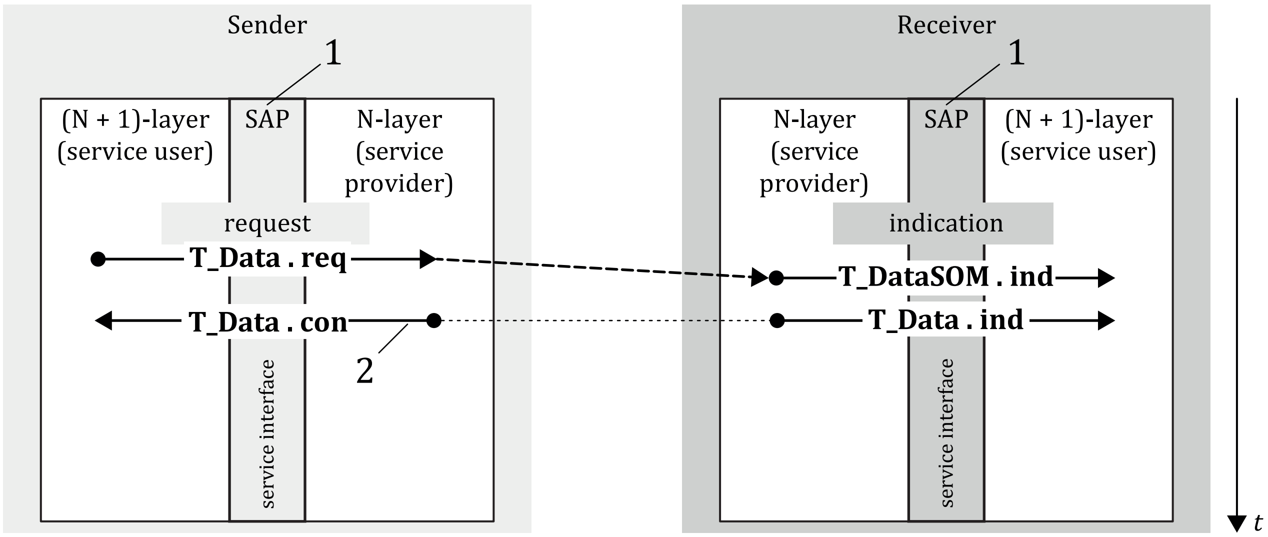
7 T_Data abstract service primitive interface definition

7.1 T_Data services

The Open Systems Interconnection (OSI) defines abstract service primitives (ASP) as an implementation-independent description of an interaction between a service user and a service provider. The abstract service primitives are defined for a particular service transition.

The ASP interface defines the service and parameter mappings beween OSI layers.

Figure 3 shows the `T_Data.req` (request), `T_Data.ind` (indication), `T_DataSOM.ind` (indication), and `T_Data.con` (confirmation) service interface.



Key

t	time
1	service access point
2	read back from N-layer service provider

Figure 3 — T_Data.req, T_Data.ind, T_DataSOM.ind and T_Data.con service interface

7.2 T_Data interface

The ASP `T_Data` interface is independent (abstraction) of the transport protocol used in the OSI-4 layer. It connects the session layer (OSI-5) and the various transport layers (OSI-4).

The ASP `T_Data` interface shall support the services as specified in Table 3.

Table 3 — ASP `T_Data` interface

ASP	Description
`T_Data.req`	This service is used by the `T_Data` interface to request the transfer of a message.
`T_Data.ind`	This service is used to signal to the `T_Data` interface the completion of a message reception.
`T_DataSOM.ind`	This service is used to signal to the `T_Data` interface the beginning of a segmented message reception.
`T_Data.con`	This service confirms to the `T_Data` interface that the requested service has been carried out (successfully or not).

7.3 Data type definitions

This requirement specifies the data types of the abstract service primitive interface parameters.

The data types shall be in accordance to:

— `Enum` = 8-bit enumeration;

— `Unsigned Byte` = 8-bit unsigned numeric value;

— `Unsigned Word =` 16-bit unsigned numeric value;

— `Unsigned Long =` 32-bit unsigned numeric value;

— `Byte Array` = sequence of 8-bit aligned data;

— `Word Array` = sequence of 16-bit aligned data;

— `Bit String =` 8-bit binary coded.

8 Transport and network layer services

8.1 General

In order to describe the functioning of the transport and network layer, it is necessary to consider services provided to higher layers and the internal operation of the transport and network layer.

All transport and network layer services have the same general structure. To define the services, three types of service primitive are specified:

— a service request primitive, used by higher communication layers or the application to pass control information and data required to be transmitted to the network layer;

— a service indication primitive, used by the network layer to pass status information and received data to upper communication layers or the application;

— a service confirmation primitive, used by the network layer to pass status information to higher communication layers or the application.

This service specification does not specify an application programming interface but only a set of service primitives that are independent of any implementation.

All transport and network layer services have the same general format. Service primitives are written in the form:

```
service_name.type          (
                           parameter A,
                           parameter B
                           [,parameter C, ...]
                           )
```

where "service_name" is the name of the service, e.g. TL_/NL_Data, "type" indicates the type of service primitive, and "parameter A, parameter B [,parameter C, ...]" are the TL_/NL_SDU as a list of values passed by the service primitive. The square brackets indicate that this part of the parameter list is optional.

The service primitives define how a service user (e.g. diagnostic application) cooperates with a service provider (e.g. network layer). The following service primitives are specified in this document: request, indication and confirm.

— Using the service primitive request (`service_name.req`), a service user requests a service from the service provider.

— Using the service primitive indication (`service_name.ind`), the service provider informs a service user about an internal event of the network layer or the service request of a peer protocol layer entity service user.

— With the service primitive confirm (`service_name.con`), the service provider informs the service user about the result of a preceding service request of the service user.

The service interface defines a set of services that are needed to access the functions offered by the transport and network layer, i.e. transmission/reception of data and setting of protocol parameters.

Two types of service are defined:

a) communication services: these services, of which the following are defined, enable the transfer of up to 4 294 967 295 bytes of data:

   1) `Data.req`: this service is used to request the transfer of data. If necessary, the network layer segments the data;

   2) `Data_FF.ind`: this service is used to indicate the beginning of a segmented message reception to the upper layer;

   3) `Data.ind`: this service is used to provide received data to the higher layers;

   4) `Data.con`: this service confirms to the higher layers that the requested service has been carried out (successfully or not);

b) protocol parameter setting services: these services, of which the following are defined, enable the dynamic setting of protocol parameters:

   1) `ChangeParameter.req`: this service is used to request the dynamic setting of specific internal parameters;

   2) `ChangeParameter.con`: this service confirms to the upper layer that the request to change a specific protocol has completed (successfully or not).

8.2 Transport and network layer abstract service primitives

8.2.1 Data.req

The service primitive requests transmission of < `MessageData` > with < `Length` > bytes from the sender to the receiver peer entities identified by the address information in `SA`, `TA`, `TAtype` [and `AE`] (see 8.3 for parameter definition).

```
Data.req          (
                  Mtype
                  SA
                  TA
                  TAtype
                  [AE]
                  <MessageData>
                  <Length>
                  )
```

Each time the `Data.req` service is called, the transport and network layer shall indicate the completion (or failure) of the message transmission to the service user by issuing a `Data.con` service call.

8.2.2 Data.con

The `Data.con` service is issued by the transport and network layer. The service primitive confirms the completion of a `Data.req` service identified by the address information in `SA`, `TA`, `TAtype` [and `AE`]. The parameter < `Result` > provides the status of the service request (see 8.3 for parameter definition).

```
Data.con            (
                    Mtype
                    SA
                    TA
                    TAtype
                    [AE]
                    <Result>
                    )
```

8.2.3 Data_FF.ind

The `Data_FF.ind` service is issued by the transport and network layer. The service primitive indicates to the adjacent upper layer the arrival of a FirstFrame (FF) of a segmented message received from a peer protocol entity, identified by the address information in `SA`, `TA`, `TAtype` [and `AE`] (see 8.3 for parameter definition). This indication shall take place upon receipt of the FF of a segmented message.

```
Data_FF.ind            (
                       Mtype
                       SA
                       TA
                       TAtype
                       [AE]
                       <Length>)
```

The `Data_FF.ind` service shall always be followed by a `Data.ind` service call from the transport and network layer, indicating the completion (or failure) of message reception.

A `Data_FF.ind` service call shall only be issued by the transport and network layer if a correct FF message segment has been received.

If the transport and network layer detect any type of error in an FF, then the message is ignored by the transport and network layer and no `Data_FF.ind` is issued to the adjacent upper layer.

If the transport and network layer receive an FF with a data length value (FF_DL) that is greater than the available receiver buffer size, then this is considered as an error condition and no `Data_FF.ind` is issued to the adjacent upper layer.

8.2.4 Data.ind

The `Data.ind` service is issued by the transport and network layer. The service primitive indicates < `Result` > events and delivers < `MessageData` > with < `Length` > bytes received from a peer protocol entity identified by the address information in `SA`, `TA`, `TAtype` [and `AE`] to the adjacent upper layer (see 8.3 for parameter definition).

The parameters < `MessageData` > and < `Length` > are valid only if < `Result` > equals OK.

```
Data.ind                (
                        Mtype
                        SA
                        TA
                        TAtype
                        [AE]
                        <MessageData>
                        <Length>
                        <Result>
                        )
```

The `Data.ind` service call is issued after reception of a SingleFrame (SF) message or as an indication of the completion (or failure) of a segmented message reception.

If the transport and network layer detect any type of error in an SF, then the message is ignored by the transport and network layer and no `Data.ind` is issued to the adjacent upper layer.

8.2.5 ChangeParameter.req

The service primitive is used to request the change of an internal parameter's value on the local protocol entity. The < `Parameter_Value` > is assigned to the < `Parameter` > (see 8.3 for parameter definition).

A parameter change is always possible, except after reception of the FF (`Data_FF.ind`) and until the end of reception of the corresponding message (`Data.ind`).

```
ChangeParameter.req     (
                        Mtype
                        SA
                        TA
                        TAtype
                        [AE]
                        <Parameter>
                        <Parameter_Value>
                        )
```

This is an optional service that can be replaced by fixed parameter values.

8.2.6 ChangeParameter.con

The service primitive confirms completion of a `ChangeParameter.con` service applying to a message identified by the address information in `SA`, `TA`, `TAtype` [and `AE`] (see 8.3 for parameter definition).

```
ChangeParameter.con     (
                        Mtype
                        SA
                        TA
                        TAtype
                        [AE]
                        <Parameter>
                        <Result_ChangeParameter>
                        )
```

8.3 Service data unit specification

8.3.1 Mtype, message type

Type: enumeration

Range: diagnostics, remote diagnostics

Description: the parameter Mtype is used to identify the type and range of address information parameters included in a service call. This document specifies a range of two values for this parameter. The intention is that users of this document can extend the range of values by specifying other types and combinations of address information parameters to be used with the transport and network layer protocol specified in this document. For each such new range of address information, a new value for the Mtype parameter is specified to identify the new address information.

Requirements:

— If Mtype = diagnostics, then the address information (`AI`) shall consist of the parameters `SA`, `TA` and `TAtype`.

— If Mtype = remote diagnostics, then the address information (`AI`) shall consist of the parameters `SA`, `TA`, `TAtype` and `AE`.

8.3.2 AI, address information

8.3.2.1 AI description

These parameters refer to addressing information. As a whole, the AI parameters are used to identify the source address (`SA`), the target address (`TA`) of message senders and recipients, as well as the communication model for the message (`TAtype`) and the optional address extension (`AE`).

8.3.2.2 SA, source address

Type: 8 bits

Range: 00_{16} to FF_{16}

Description: the `SA` parameter is used to encode the sending network layer protocol entity.

8.3.2.3 TA, target address

Type: 8 bits

Range: 00_{16} to FF_{16}

Description: the `TA` parameter is used to encode one or multiple (depending on the `TAtype`: `physical` or `functional`) receiving network layer protocol entities.

8.3.2.4 TAtype, target address type

Type: enumeration

Range: see Table 4

Description: the parameter `TAtype` is an extension to the TA parameter. It is used to encode the communication model used by the communicating peer entities of the transport and network layer. The following requirements are supported:

— the transport and network layer protocol is capable of carrying out parallel transmission of different messages that are not mapped onto the same `NL_AI`;

— error handling for unexpected PDUs only pertains to messages with the same AI:

  — CAN CC frames will not cause a CAN FD message to be terminated and vice-versa;

  — this explicitly prevents mixing CAN FD/CAN CC frame types in a single message.

Table 4 specifies the allowed combinations of `TAtype` communication models.

Table 4 — Allowed combinations of TAtype communication models

TAtype	Physical/Functional addressing	<Format>
TAtype #1	Physical[a]	CAN base format (CAN CC, 11-bit)
TAtype #2	Functional[b]	
TAtype #3	Physical[a]	CAN FD base format (CAN FD, 11-bit)
TAtype #4	Functional[b]	
TAtype #5	Physical[a]	CAN extended format (CAN CC, 29-bit)
TAtype #6	Functional[b]	
TAtype #7	Physical[a]	CAN FD extended format (CAN FD, 29-bit)
TAtype #8	Functional[b]	

[a] Physical addressing (1 to 1 communication) is supported for all types of network layer messages.

[b] Functional addressing (1 to *n* communication) shall only be supported for SingleFrame transmission.

Figure 4 shows an example of an enhanced diagnostic tool CAN CC request for normal addressing (TAtype #2).

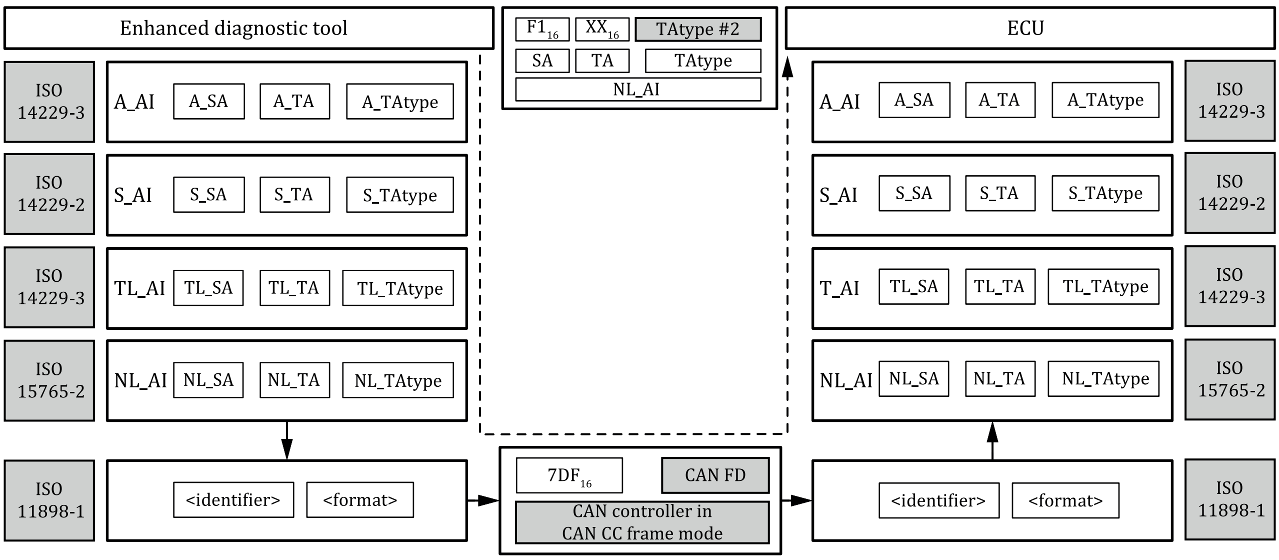


Figure 4 — Example of enhanced diagnostic tool CAN CC request for normal addressing (TAtype #2)

Figure 5 shows an example of an enhanced diagnostic tool CAN FD request for normal addressing (TAtype #4).

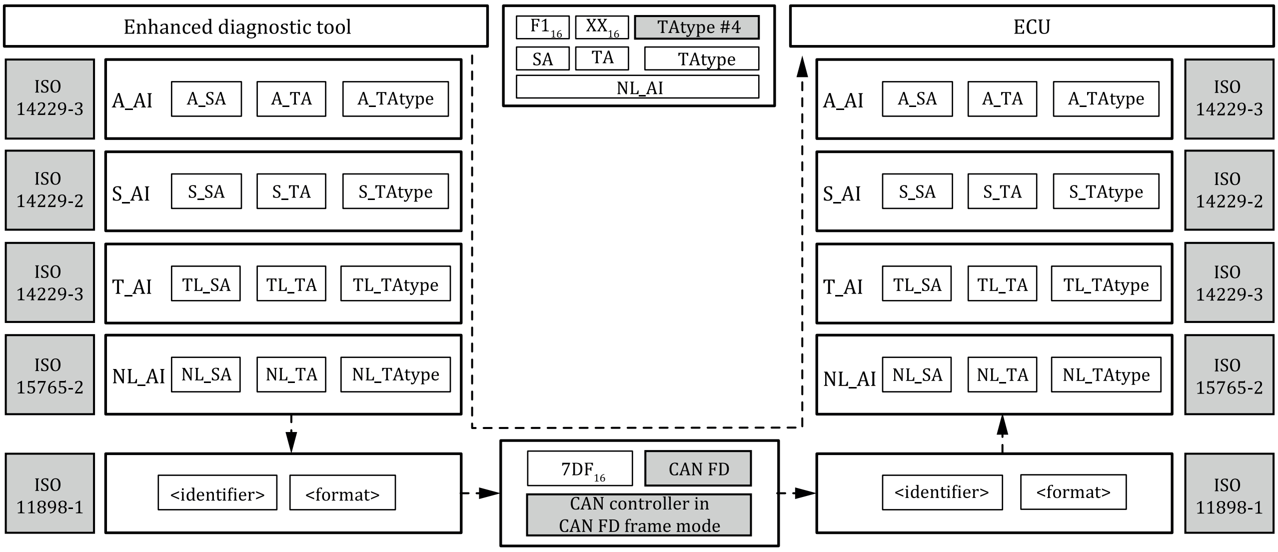


Figure 5 — Example of enhanced diagnostic tool CAN FD request for normal addressing (TAtype #4)

8.3.2.5 AE, network address extension

Type: 8 bits

Range: 00_{16} to FF_{16}

Description: the NL_AE parameter is used to extend the available address range for large networks and to encode both sending and receiving network layer entities of sub-networks other than the local network where the communication takes place. NL_AE is only part of the addressing information if Mtype is set to remote diagnostics.

8.3.3 <Length>

Type: 32 bits

Range: $0000\ 0001_{16}$ to $FFFF\ FFFF_{16}$

Description: this parameter includes the length of data to be transmitted/received.

8.3.4 <MessageData>

Type: string of bytes

Range: not applicable

Description: this parameter includes all data that the higher-layer entities exchange.

8.3.5 <Parameter>

Type: enumeration

Range: $C_{\mathrm{TL\_FCSTmin}}$, $C_{\mathrm{TL\_FCBS}}$

Description: these parameters identify parameters of the transport layer.

8.3.6 <Parameter_Value>

Type: 8 bits

Range: 00_{16} to FF_{16}

Description: this parameter is assigned to a protocol parameter <Parameter> as indicated in 9.6.5.3 and 9.6.5.4.

8.3.7 <Result>

Type: enumeration

Range: $C_{TL\_BUFFER\_OVFLW}$, C_{TL_ERROR}, $C_{TL\_INVALID\_FS}$, C_{TL_OK}, $C_{TL\_TIMEOUT\_A}$, $C_{TL\_TIMEOUT\_Bs}$, $C_{TL\_TIMEOUT\_Cr}$, $C_{TL\_UNEXP\_PDU}$, $C_{TL\_WRONG\_SN}$, $C_{TL\_WFT\_OVRN}$

Description: this parameter contains the status relating to the outcome of a service execution. If two or more errors are discovered at the same time, then the transport and network layer entities shall use the parameter value found first in this list when indicating the error to the higher layers.

— $C_{TL\_BUFFER\_OVFLW}$

This value is issued to the service user upon receipt of a FlowControl (FC) TL_PDU with FlowStatus = OVFLW. It indicates that the buffer on the receiver side of a segmented message transmission cannot store the number of bytes specified by the FirstFrame DataLength (FF_DL) parameter in the FirstFrame and therefore, the transmission of the segmented message was aborted. It can be issued to the service user on the sender side only.

— C_{TL_ERROR}

This is the general error value. It is issued to the service user when an error has been detected by the network layer and no other parameter value can be used to better describe the error. It can be issued to the service user on both the sender and receiver sides.

— $C_{TL\_INVALID\_FS}$

This value is issued to the service user when an invalid or unknown FlowStatus value has been received in a FlowControl (FC) NL_PDU; it can be issued to the service user on the sender side only.

— C_{TL_OK}

This value means that the service execution has been completed successfully; it can be issued to a service user on both the sender and receiver sides.

— $C_{TL\_TIMEOUT\_A}$

This value is issued to the protocol user when the timer t_{TL_Ar}/t_{TL_As} has passed its time-out value $C_{TL\_TIMEOUT\_Asmax}/C_{TL\_TIMEOUT\_Armax}$; it can be issued to service users on both the sender and receiver sides.

— $t_{TL\_TIMEOUT\_Bs}$

This value is issued to the service user when the timer t_{TL_Bs} has passed its time-out value $t_{TL\_TIMEOUT\_Bs}$; it can be issued to the service user on the sender side only.

— $t_{TL\_TIMEOUT\_Cr}$

This value is issued to the service user when the timer t_{TL_Cr} has passed its time-out value $t_{TL\_TIMEOUT\_Cr}$; it can be issued to the service user on the receiver side only.

— $C_{TL\_UNEXP\_PDU}$

This value is issued to the service user upon receipt of an unexpected protocol data unit; it can be issued to the service user on the receiver side only.

— $C_{TL\_WFT\_OVRN}$

This value is issued to the service user when the receiver has transmitted C_{TL_WFTmax} FlowControl TL_PDUs with FlowStatus = WAIT in a row and following this, it cannot meet the performance requirement for the transmission of a FlowControl TL_PDU with FlowStatus = CTS. It can be issued to the service user on the receiver side only.

— $C_{TL\_WRONG\_SN}$

This value is issued to the service user upon receipt of an unexpected SequenceNumber (PCI.SN) value; it can be issued to the service user on the receiver side only.

8.3.8 <Result_ChangeParameter>

Type: enumeration

Range: C_{TL_OK}, $C_{TL\_RX\_ON}$, $C_{TL\_WRONG\_PARAMETER}$, $C_{TL\_WRONG\_VALUE}$

Description: this parameter contains the status relating to the outcome of a service execution.

— C_{TL_OK}

This value means that the service execution has been completed successfully; it can be issued to a service user on both the sender and receiver sides.

— $C_{TL\_RX\_ON}$

This value is issued to the service user to indicate that the service did not execute since reception of the message identified by `<AI>` was taking place; it can be issued to the service user on the receiver side only.

— $C_{TL\_WRONG\_PARAMETER}$

This value is issued to the service user to indicate that the service did not execute due to an undefined `<Parameter>`; it can be issued to a service user on both the receiver and sender sides.

— $C_{TL\_WRONG\_VALUE}$

This value is issued to the service user to indicate that the service did not execute due to an out-of-range `<Parameter_Value>`; it can be issued to a service user on both the receiver and sender sides.

8.4 ASP T_Data to TL_Data interface parameter mapping

The transport layer ASP interface specifies a set of services that are needed to access the functions offered by the `T_Data` interface, i.e. transmission/reception of data parameters. The `T_Data` interface is independent of the underlying transport layer protocol used in the OSI-4 layer.

The transport layer protocol shall support the mapping of the ASP `T_Data` parameters to the `TL_Data` interface as specified in Table 5.

Table 5 — ASP T_Data to TL_Data interface parameter mapping

T_Data	TL_Data	.req	.ind	.con	Description
`T_Mtype`	`TL_Mtype`	X	X	X	packet transport protocol type
`T_AI[TAtype]`	`TL_AI[TAtype]`	X	X	X	address information [target address type]
`T_AI[SA]`	`TL_AI[SA]`	X	X	X	address information [source address]
`T_AI[TA]`	`TL_AI[TA]`	X	X	X	address information [target address] to be added to PDU if `T_AI[TAtype]` = `DiagExtAddr`
`T_AI[AE]`	`TL_AI[AE]`	X	X	X	address information [address extension] to be added to PDU if `T_AI[TAtype]` = `RDiagMixAddr`
`T_PDU_Length`	`TL_PDU_Length`	X	X	–	`T_PDU_Length` = length of `A_PDU`
`T_Data`	`TL_Data`	X	X	–	`TL_Data` field of the `TL_PDU`
`T_Result`	`TL_Result`	–	X	X	result

Key

X supported

— not supported

9 Transport layer protocol

9.1 Protocol functions

This document specifies an unconfirmed transport layer communication protocol for the exchange of data between network nodes, e.g. from ECU to ECU, or between external test equipment and an ECU. If the data to be transferred does not fit into a single CAN frame, a segmentation method is provided.

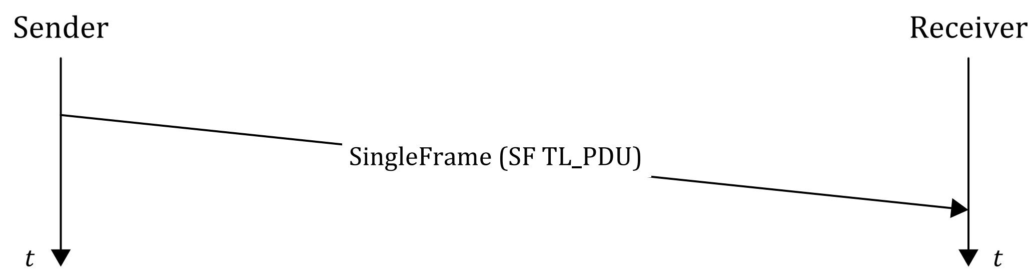
The transport layer protocol performs the following functions:

— transmission/reception of messages up to 4 294 967 295 (2^{32} – 1) data bytes;

— reporting of transmission/reception completion (or failure).

9.2 Single frame message transmission

Figure 6 shows an example of an unsegmented message transmission.



Key

t time

Figure 6 — Example of an unsegmented message

9.3 Multiple frame message transmission

Transmission of longer messages is performed by segmenting the message and transmitting multiple TL_PDUs. Reception of longer messages is performed by receiving multiple TL_PDUs and reassembling of received data bytes (concatenation). The multiple TL_PDUs are called FirstFrame (for the first TL_PDU of the message) and ConsecutiveFrame (for all the following TL_PDUs).

The receiver of a segmented TL_PDU message has the possibility of adapting the transmission throughout to its capability by means of the FlowControl mechanism, using the FlowControl protocol data units (FC TL_PDU).

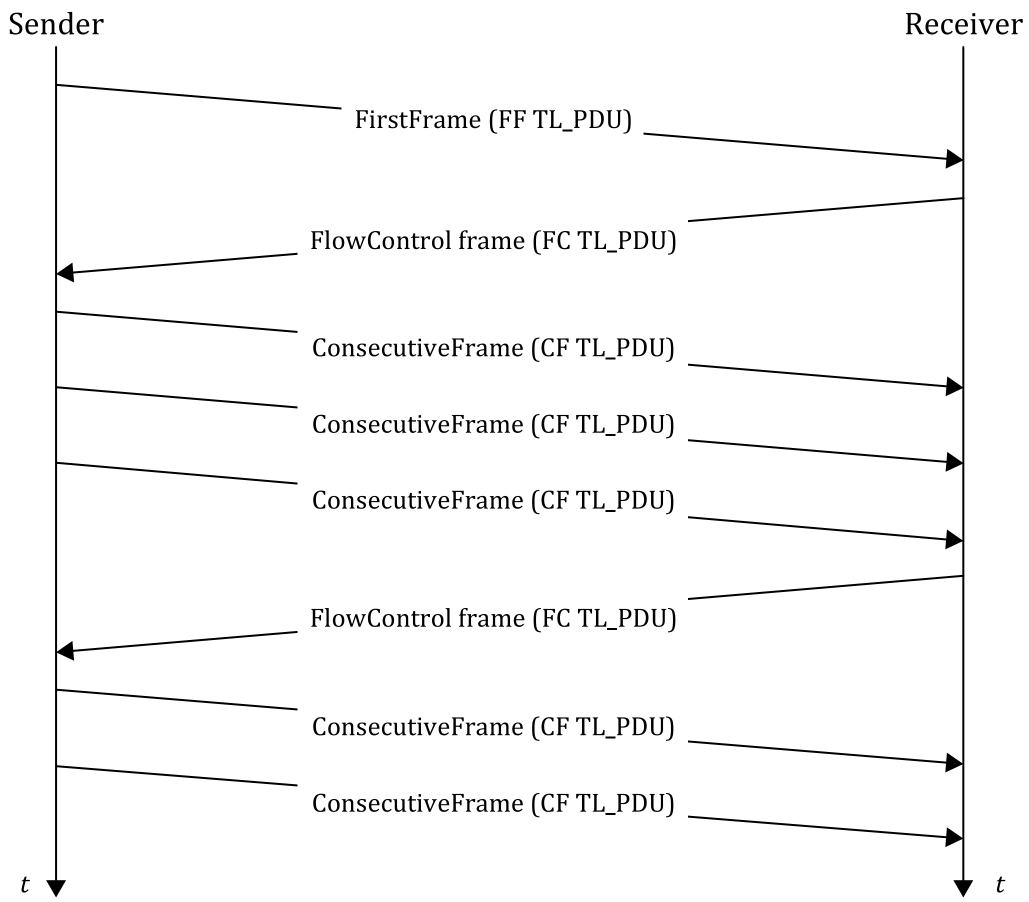
Messages that are larger than the maximum SF_DL allowed by the used TX_DL are segmented into:

— a FirstFrame protocol data unit (FF TL_PDU), containing the first set of data bytes; and

— one or more ConsecutiveFrame protocol data units (CF TL_PDU), each containing consecutive sets of data bytes. The last (or only) CF TL_PDU contains the last set of data bytes.

The message length is transmitted in the FF TL_PDU. All CF TL_PDUs are numbered (sequence number) by the sender to help the receiver reassemble them in the same order.

Figure 7 shows an example of a segmented message transmission.



Key

t time

Figure 7 — Example of a segmented message

The FlowControl FC TL_PDU is used to adjust the sending behaviour of the sender to the transport layer capabilities of the receiver.

9.4 Transport layer protocol data units

9.4.1 Protocol data unit types

The communication between the peer protocol entities of the transport layer in different nodes is done by means of exchanging TL_PDUs.

This document specifies four different types of transport layer protocol data units, SingleFrame (SF TL_PDU), FirstFrame (FF TL_PDU), ConsecutiveFrame (CF TL_PDU) and FlowControl (FC TL_PDU), which are used to establish a communication path between the peer transport layer entities, to exchange communication parameters, to transmit user data and to release communication resources.

9.4.2 SF TL_PDU

The SF TL_PDU is identified by the SingleFrame protocol control information (SF TL_PCI). The SF TL_PDU shall be sent out by the sending network entity and can be received by one or multiple receiving network entities. It shall be sent out to transfer a service data unit that can be transferred via a single service request to the network layer and to transfer unsegmented messages.

9.4.3 FF TL_PDU

The FF TL_PDU is identified by the FirstFrame protocol control information (FF TL_PCI). The FF TL_PDU shall be sent out by the sending network entity and received by a unique receiving network entity for the duration of the segmented message transmission. It identifies the first TL_PDU of a segmented message transmitted by a network sending entity. The receiving network layer entity shall start assembling the segmented message on receipt of an FF TL_PDU.

9.4.4 CF TL_PDU

The CF TL_PDU is identified by the ConsecutiveFrame protocol control information (CF TL_PCI). The CF TL_PDU transfers data segments (TL_Data) of the service data unit message data (`<MessageData>`). All TL_PDUs transmitted by the sending entity after the FF TL_PDU shall be encoded as CF TL_PDUs. The receiving entity shall pass the assembled message to the service user of the network receiving entity after the last CF TL_PDU has been received. The CF TL_PDU shall be sent out by the sending network entity and received by a unique receiving network entity for the duration of the segmented message transmission.

9.4.5 FC TL_PDU

The FC TL_PDU is identified by the FlowControl protocol control information (FC TL_PCI). The FC TL_PDU instructs a sending network entity to start, stop or resume transmission of CF TL_PDUs. It shall be sent by the receiving network layer entity to the sending network layer entity, when ready to receive more data, after correct reception of

a) an FF TL_PDU; or

b) the last CF TL_PDU of a block of ConsecutiveFrames, if further ConsecutiveFrames need to be sent.

The FC TL_PDU can also inform a sending network entity to pause transmission of CF TL_PDUs during a segmented message transmission or to abort the transmission of a segmented message if the length information (FF_DL) in the FF TL_PDU transmitted by the sending entity exceeds the buffer size of the receiving entity.

9.4.6 TL_PDU field specification

9.4.6.1 TL_PDU format

The protocol data unit (TL_PDU) enables the transfer of data between the network layer in one node and the network layer in one or more other nodes (peer protocol entities). All TL_PDUs consist of two fields, as given in Table 6.

Table 6 — TL_PDU format

Protocol control information	Data field
TL_PCI	A_PDU

9.4.6.2 Protocol control information (TL_PCI)

The TL_PCI field identifies the type of TL_PDUs exchanged. It is also used to exchange other control parameters between the communicating transport layer entities.

NOTE For a detailed specification of all TL_PCI parameters, see 9.6.

9.4.6.3 Data field (A_PDU)

The A_PDU in the TL_PDU is used to transmit the service user data received in the `<MessageData>` parameter in the TL_Data.req service call. The `<MessageData>`, if needed, is segmented into smaller parts that each fit into the TL_PDU data field before they are transmitted over the network.

The size of data field depends on the TL_PDU type and the value of TX_DL.

9.5 Transmit data length (TX_DL) configuration

9.5.1 Definition of TX_DL configuration values

The transmit data length (TX_DL) configures the maximum usable payload length of the data link layer for the application that implements the transport layer as specified in this document. The TX_DL value is defined as the real payload length in bytes to provide simple calculations and sanity checks for the length definitions for TL_PCI types specified in 9.6. The valid TX_DL values are derived from the payload length for data length code (DLC) values from 8 to 15 (see ISO 11898-1).

Table 7 describes valid transmit data length (TX_DL) values.

Table 7 — Definition of TX_DL configuration values

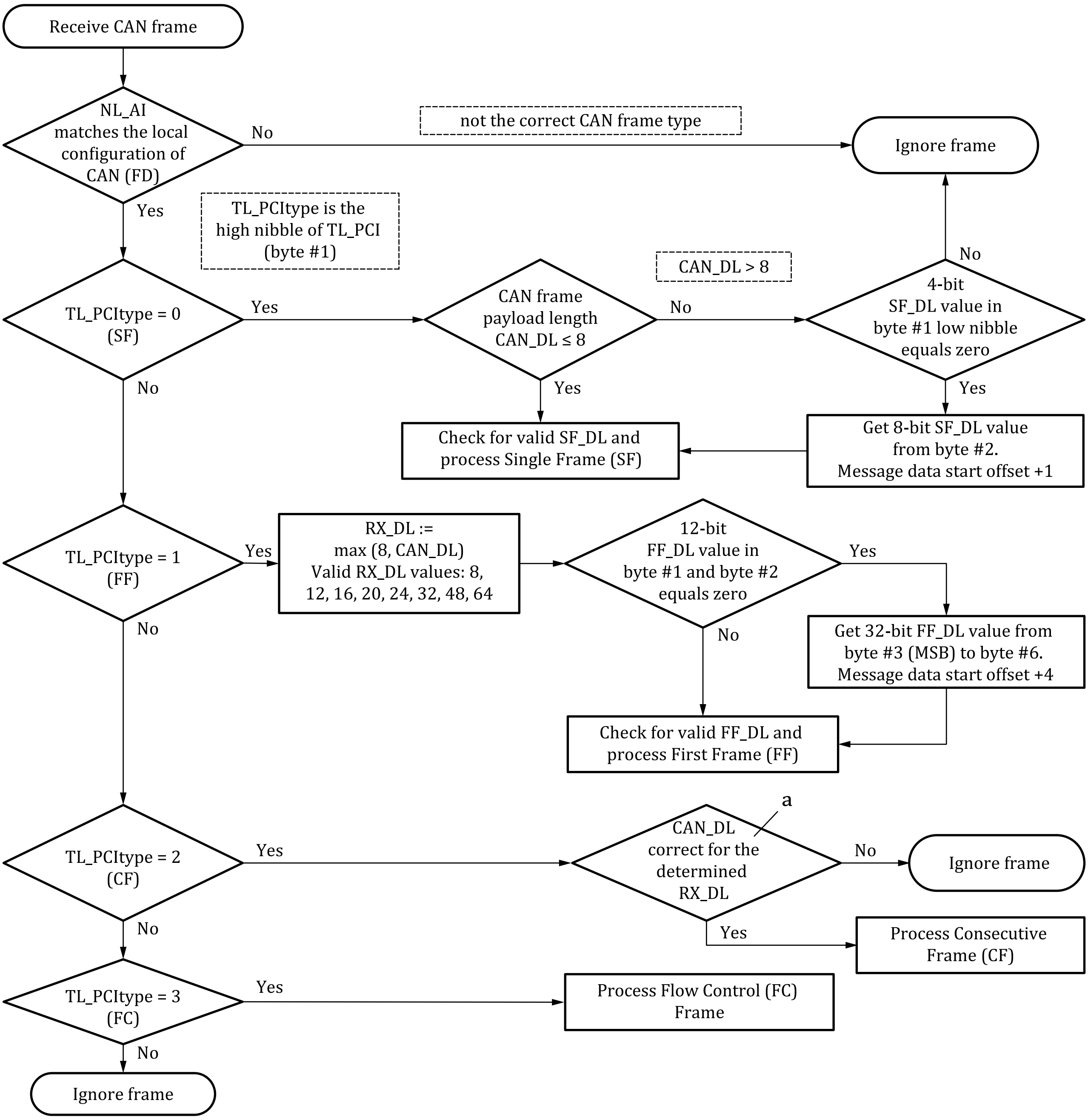
TX_DL	Description
< 8	Invalid This range of values is invalid.
= 8	Configured CAN frame maximum payload length of 8 byte For the use with ISO 11898-1 CAN CC type frames and CAN FD type frames: — Valid DLC value range: 2 to 8; — Valid CAN_DL value range: 2 to 8.
> 8	Configured CAN frame maximum payload length greater than 8 byte For the use with ISO 11898-1 CAN FD type frames only: — Valid DLC value range: 2 to 15; — Valid CAN_DL value range: 2 to 8, 12, 16, 20, 24, 32, 48, 64; — Valid TX_DL value range: 12, 16, 20, 24, 32, 48, 64; — CAN_DL ≤ TX_DL.

9.5.2 Verifying the correctness of received CAN frames

Due to the fact that the TX_DL configuration of the sending node is not known by the receiver, the receiving node shall always adapt to the TX_DL settings of the sender.

The locally configured TL_TAtype allows checking the received CAN frame type (CAN CC/CAN FD) and is used to ignore wrong TL_TAtype frames. Once the TL_TAtype is correct, the different TL_PCItype values can be checked and assumptions on the RX_DL (the transmitters TX_DL) can be made.

Figure 8 shows how to process incoming CAN frames.



Key

a CAN_DL shall be correct if the value matches RX_DL for all CF TL_PDUs except for the last (or only) CF TL_PDU; the last (or only) CF TL_PDU shall pass this check if CAN_DL is less or equal than RX_DL and the requirements in 9.6.4.2 are met; RX_DL comes from the FF TL_PDU and is fixed for this TL_PDU reception process.

Figure 8 — Verifying received CAN frames

9.5.3 Receiver determination RX_DL

To determine the RX_DL from a received FF TL_PDU, the payload length in bytes (CAN_DL) is used.

— For CAN_DL values less than 8 bytes, the RX_DL value is invalid.

NOTE A valid FF TL_PDU always has a CAN_DL value greater than or equal to 8.

— For CAN_DL values equal to 8 bytes, the RX_DL value shall be 8.

— For CAN_DL values greater than 8 bytes, the RX_DL value equals the CAN_DL value.

Table 8 specifies the received CAN_DL to RX_DL mapping table.

Table 8 — Received CAN_DL to RX_DL mapping table

Received CAN_DL	RX_DL
0 to 7	invalid
8	8
12	12
16	16
20	20
24	24
32	32
48	48
64	64

9.6 Protocol control information specification

9.6.1 TL_PCI

Each TL_PDU is identified by means of a TL_PCI. See Table 9 and Table 10.

Table 9 specifies the TL_PCItype bit values.

Table 9 — Definition of TL_PCItype bit values

TL_PCI byte #1 bits 7 to 4	Description
0000_2 (0_{16})	**SingleFrame (SF TL_PDU)** For unsegmented messages with CAN_DL ≤ 8, the message length is embedded in lower nibble of the only PCI byte (byte #1). For unsegmented messages with CAN_DL > 8, the SingleFrame escape sequence shall be used where the lower nibble of the first PCI byte (byte #1) is set to 0000_2 and the message length is embedded in the second PCI byte (byte #2). SingleFrame (SF) shall be used to support the transmission of messages that can fit in a single CAN frame.
0001_2 (1_{16})	**FirstFrame (FF TL_PDU)** A FirstFrame (FF) shall only be used to support the transmission of messages that cannot fit in a single CAN frame, i.e. segmented messages. On receipt of a FirstFrame (FF), the receiving network layer entity shall start assembling the segmented message. — For segmented messages with a message length ≤ 4 095, the lower nibble of the first PCI byte (byte #1) and the second PCI byte (byte #2) includes the message length. — For segmented messages with a message length > 4 095, the FirstFrame escape sequence shall be used where the lower nibble of the first PCI byte (byte #1) is set to 0000_2 and the second PCI byte (byte #2) is set to zero. The message length is embedded in the following four PCI bytes (byte #3 to byte #6, MSB first).
0010_2 (2_{16})	**ConsecutiveFrame (CF TL_PDU)** When sending segmented data, all consecutive frames following the FF are encoded as ConsecutiveFrame (CF). On receipt of a ConsecutiveFrame (CF), the receiving network layer entity shall assemble the received data bytes until the whole message is received. The receiving entity shall pass the assembled message to the adjacent upper protocol layer after the last frame of the message has been received without error.

Table 9 *(continued)*

TL_PCI byte #1 bits 7 to 4	Description
0011_2 (3_{16})	**FlowControl (FC TL_PDU)** The purpose of FlowControl (FC) is to regulate the rate at which CF TL_PDUs are sent to the receiver. Three distinct types of FlowControl (FC) protocol control information are specified to support this function. The type is indicated by a field of the protocol control information called FlowStatus (FS), as specified in 9.6.5.1.
4_{16} to F_{16}	**Reserved** This range of values is reserved by this document.

Table 10 shows a summary of TL_PCI bytes.

Table 10 — Summary of TL_PCI bytes

TL_PDU name	TL_PCI bytes						
	Byte #1		Byte #2	Byte #3	Byte #4	Byte #5	Byte #6
	Bits 7 to 4	Bits 3 to 0					
SingleFrame (SF) (CAN_DL ≤ 8)	0000_2	SF_DL	—	—	—	—	—
SingleFrame (SF) (CAN_DL > 8)[a]	0000_2	0000_2	SF_DL	—	—	—	—
FirstFrame (FF) (8 < FF_DL ≤ 4 095)	0001_2	FF_DL		—	—	—	—
FirstFrame (FF) (4 095 < FF_DL ≤ 4 294 967 295)[b]	0001_2	0000_2	$0000\ 0000_2$	FF_DL			
ConsecutiveFrame (CF)	0010_2	C_{TL_CFSN}	—	—	—	—	—
FlowControl (FC)	0011_2	C_{TL_FCFS}	C_{TL_FCBS}	$C_{TL\_FCSTmin}$	N/A	N/A	N/A

[a] Messages with CAN_DL > 8 shall use an escape sequence where the lower nibble of byte #1 is set to 0 (invalid length). This indicates to the transport layer that the value of SF_DL is determined based on the next byte in the frame (byte #2). As CAN_DL is specified to be greater than 8, this definition is only valid for CAN FD type frames.

[b] Messages larger than 4 095 bytes shall use an escape sequence where the lower nibble of byte #1 and all bits in byte #2 are set to 0 (invalid length). This indicates to the transport layer that the value of FF_DL is determined based on the next 32 bits in the frame (byte #3 is the MSB and byte #6 the LSB).

NOTE Byte numbers with dash lines are not utilized for PCI information, but depending on the TL_PDU, it is possible that they are utilized for payload data.

9.6.2 SingleFrame TL_PCI parameter definition

9.6.2.1 SF TL_PCI byte

The parameter SingleFrame data length (SF_DL) is used in the SF TL_PDU to specify the number of service message data bytes. The ranges of valid SF_DL values depend on the configured transmit data link layer data length (TX_DL) and the actual payload to be transmitted (see Table 11 and Table 12). If the value of TX_DL > 8 and the payload size results in CAN_DL exceeding 8, then bits 0 to 3 of the first PCI byte (byte #1) are set to 0 and the SF_DL is embedded in the second PCI byte (byte #2); see Table 11.

Table 11 — Definition of SF_DL values with CAN_DL ≤ 8

Value bits 7 to 4	Description
0000_2	**Reserved** This value is reserved by this document.
0001_2 to 0110_2	**SingleFrame DataLength (SF_DL)** SF_DL shall be assigned the value of the service parameter `<Length>`.
0111_2	**SingleFrame DataLength (SF_DL) with normal addressing only** SF_DL shall be assigned the value of the service parameter `<Length>`. SF_DL = 7 is only allowed with normal addressing.
other values	**Invalid** This range of values is invalid.

NOTE 1 SF_DL is encoded in the low nibble of first TL_PCI byte (byte #1) value.

Table 12 — Definition of SF_DL values (CAN_DL > 8)

Value	Description
$0000\ 0000_2$ to $0000\ 0110_2$	**Reserved** This value is reserved by this document.
$0000\ 0111_2$	**SingleFrame DataLength (SF_DL) with extended addressing or mixed addressing** SF_DL shall be assigned the value of the service parameter `<Length>`. SF_DL = 7 is only allowed with extended addressing or mixed addressing.
$0000\ 1000_2$ to (CAN_DL – 3)	**SingleFrame DataLength (SF_DL)** SF_DL shall be assigned the value of the service parameter `<Length>`.
(CAN_DL – 2)	**SingleFrame DataLength (SF_DL) with normal addressing only** SF_DL shall be assigned the value of the service parameter `<Length>`. SF_DL = (CAN_DL – 2) is only allowed with normal addressing.
other values	**Invalid** This range of values is invalid.

NOTE 2 SF_DL is encoded in the second TL_PCI byte (byte #2) value. This is only allowed for CAN FD type frames.

9.6.2.2 SF_DL error handling

Received CAN_DL is less or equal to 8:

— if the transport layer receives an SF TL_PDU, where SF_DL is equal to 0, then the transport layer shall ignore the received SF TL_PDU;

— if the transport layer receives an SF TL_PDU, where SF_DL is greater than (CAN_DL – 1) of the received frame when using normal addressing or greater than (CAN_DL – 2) of the received frame for extended or mixed addressing, then the transport layer shall ignore the received SF TL_PDU;

— in the case of CAN frame data padding (see 11.3.2.1), if the transport layer receives an SF TL_PDU, where the CAN_DL does not equal to 8, then the transport layer shall ignore the received SF TL_PDU;

— in the case of CAN frame data optimization (see 11.3.2.2), if the transport layer receives an SF TL_PDU, where the value of SF_DL does not match the valid values shown in Table 13, then the transport layer shall ignore the received SF TL_PDU.

Table 13 — Allowed SF_DL values for a given addressing scheme with optimized CAN_DL

Addressing type	CAN_DL value							
	0 to 1	2	3	4	5	6	7	8
Normal	Invalid	SF_DL = 1	SF_DL = 2	SF_DL = 3	SF_DL = 4	SF_DL = 5	SF_DL = 6	SF_DL = 7
Mixed or extended	Invalid	Invalid	SF_DL = 1	SF_DL = 2	SF_DL = 3	SF_DL = 4	SF_DL = 5	SF_DL = 6

Received CAN_DL is greater than 8:

— if the transport layer receives an SF TL_PDU, where the low nibble of the first PCI byte is not 0000_2, then the transport layer shall ignore the received SF TL_PDU;

— if the transport layer receives an SF TL_PDU, where the value of SF_DL does not fall into the valid range shown in Table 14, then the transport layer shall ignore the received SF TL_PDU.

Table 14 — Allowed SF_DL values for a given CAN_DL greater than 8 and addressing scheme

Addressing type	CAN_DL value						
	12	16	20	24	32	48	64
Normal	8 ≤ SF_DL ≤ 10	11 ≤ SF_DL ≤ 14	15 ≤ SF_DL ≤ 18	19 ≤ SF_DL ≤ 22	23 ≤ SF_DL ≤ 30	31 ≤ SF_DL ≤ 46	47 ≤ SF_DL ≤ 62
Mixed or extended	7 ≤ SF_DL ≤ 9	10 ≤ SF_DL ≤ 13	14 ≤ SF_DL ≤ 17	18 ≤ SF_DL ≤ 21	22 ≤ SF_DL ≤ 29	30 ≤ SF_DL ≤ 45	46 ≤ SF_DL ≤ 61

9.6.3 FirstFrame TL_PCI parameter definition

9.6.3.1 FirstFrame DataLength (FF_DL) parameter definition

The parameter FF_DL is used in the FF TL_PDU to specify the number of service message data bytes. For the sender, the range of valid FF_DL values depends on the addressing scheme and the configured transmit data link layer data length (TX_DL). The minimum values of FF_DL (FF_DL_{min}) based on addressing scheme and TX_DL are specified in Table 15.

Table 15 — Minimum value of FF_DL based on the addressing scheme

Condition	FF_DL_{min} value
If the configured TX_DL is 8 and normal addressing is used.	8
If the configured TX_DL is 8 and mixed or extended addressing is used.	7
If the configured TX_DL > 8 and normal addressing is used.	TX_DL – 1
If the configured TX_DL > 8 and mixed or extended addressing is used.	TX_DL – 2

The receiver of an FF TL_PDU does not have knowledge of the TX_DL of the sender. The receiver determines the minimum value of FF_DL (FF_DL_{min}) from Table 16 based on the configured addressing scheme and the retrieved value of RX_DL from the CAN_DL of the FF TL_PDU (see 9.5.3 for definition of how the receiver determines RX_DL).

Only messages larger than 4 095 bytes in length shall use the escape sequence where the lower nibble of the first PCI byte (byte #1) and the entire second PCI byte (byte #2) have all bits set to 0. This tells the transport layer that the FF_DL is to be determined based on a 32-bit value contained in byte #3 (MSB) through byte #6 (LSB) of the first frame.

Table 16 — Valid FF_DL values

Value	Description
0 to (FF_DL_{min} – 1)	**Invalid** This range of values is invalid.
FF_DL_{min} to 4 095	**FirstFrame DataLength (FF_DL) without escape sequence** The encoding of the segmented message length results in a 12-bit length value (FF_DL) where the least significant bit (LSB) is specified to be bit 0 of the second TL_PCI byte (byte #2) and the most significant bit (MSB) is bit 3 of the first TL_PCI byte (byte #1). The maximum segmented message length supported is equal to 4 095 bytes of user data. It shall be assigned the value of the service parameter `<Length>`.
4 096 to 4 294 967 295 (2^{32}–1)	**FirstFrame DataLength (FF_DL) with escape sequence** The encoding of the segmented message length results in a 32-bit length value (FF_DL) where the least significant bit (LSB) is specified to be bit 0 of the sixth TL_PCI byte (byte #6) and the most significant bit (MSB) is bit 7 of the third TL_PCI byte (byte #3). The maximum segmented message length supported is equal to 4 294 967 295 bytes of user data. It shall be assigned the value of the service parameter `<Length>`.

9.6.3.2 FF_DL error handling

If the transport layer receives a TL_PDU indicating an FF TL_PDU and CAN_DL < 8, then the transport layer shall ignore the FF TL_PDU.

If the transport layer receives an FF TL_PDU with an FF_DL that is greater than the available receiver buffer size, then this shall be considered as an error condition. The transport layer shall abort the message reception and send an FC TL_PDU with the parameter FlowStatus = Overflow.

If the transport layer receives a FirstFrame with an FF_DL that is less than FF_DL_{min}, the transport layer shall ignore the received FF TL_PDU and not transmit an FC TL_PDU.

NOTE Legacy devices that only support the 12-bit version of FF_DL does not send an FC TL_PDU if the escape sequence is used as these devices would interpret FF_DL to be less than FF_DL_{min} and as such, not send an FC TL_PDU.

If an FF TL_PDU is received with the escape sequence (where all bits of the lower nibble of PCI byte #1 and all bits of PCI byte #2 are set to 0) and the FF_DL ≤ 4 095, then the transport layer shall ignore the FF TL_PDU and not transmit an FC TL_PDU.

9.6.4 ConsecutiveFrame TL_PCI parameter definition

9.6.4.1 CF TL_PCI byte

The payload data length CAN_DL of the received CAN frame shall match the RX_DL value, which is determined in the reception process of the FF TL_PDU. Only the last CF TL_PDU in the multi-frame transmission may contain less than RX_DL bytes.

9.6.4.2 Transmitter requirements for last consecutive frame

A transmitter of a multi-frame message shall send the last (or only) consecutive frame with only the required number of bytes. See examples below for clarification (normal addressing).

EXAMPLE 1 A last CF TL_PDU with 9 data bytes is sent padded to 12 bytes.

EXAMPLE 2 A last CF TL_PDU with 3 bytes of data is sent with a CAN_DL of 8 or 4 [dependent upon the use of CAN frame optimization (see 11.3.2.1 and 11.3.2.2)].

9.6.4.3 SequenceNumber (SN) parameter definition

The parameter SN is used in the CF TL_PDU (ConsecutiveFrames) to specify the following:

— the numeric ascending order of the CF TL_PDU;

— that the SN shall start with zero for all segmented messages; the FF TL_PDU shall be assigned the value zero; it does not include an explicit SequenceNumber in the TL_PCI field but shall be treated as the segment number zero;

— that the SN of the first CF TL_PDU shall be set to one;

— that the SN shall be incremented by one for each new CF TL_PDU that is transmitted during a segmented message transmission;

— that the SN value shall not be affected by any FC TL_PDU;

— that when the SN reaches the value of 15, it shall wraparound and be set to zero for the next CF TL_PDU.

This shall lead to the sequence given in Table 17.

Table 17 — Summary of SN definition

TL_PDU	FF	CF	CF	CF	CF	CF	CF	CF
SN	0_{16}	1_{16}	…	E_{16}	F_{16}	0_{16}	1_{16}	…

See Table 18 for a definition of SN values.

Table 18 — Definition of SN values

Value	Description
0_{16} to F_{16}	**SequenceNumber ($C_{\mathrm{TL\_CFSN}}$)** The SequenceNumber (SN) shall be encoded in the lower nibble bits of TL_PCI byte #1. The $C_{\mathrm{TL\_CFSN}}$ shall be set to a value within the range of 0_{16} to F_{16}.

9.6.4.4 SequenceNumber (SN) error handling

If a CF TL_PDU message is received with an unexpected SequenceNumber not in accordance with the definition in 9.6.4.3, the message reception shall be aborted and the transport layer shall initiate a TL_Data.ind service call with the parameter <TL_Result> = $C_{\mathrm{TL\_WRONG\_SN}}$ (wrong sequence number) to the adjacent upper layer.

9.6.5 FlowControl TL_PCI parameter definition

9.6.5.1 FlowStatus (FS) parameter definition

The parameter FlowStatus ($C_{\mathrm{TL\_FCFS}}$) indicates whether the sending network entity can proceed with the message transmission.

A sending network entity shall support all specified (not reserved) values of the $C_{\mathrm{TL\_FCFS}}$ parameter.

Table 19 specifies the $C_{\mathrm{TL\_FCFS}}$ values.

Table 19 — Definition of C_{TL_FCFS} values

Value	Description
0_{16}	**ContinueToSend ($C_{TL\_FCFS(CTS)}$)** The FlowControl ContinueToSend parameter shall be encoded by setting the lower nibble of the TL_PCI byte #1 to "0". It shall cause the sender to resume the sending of ConsecutiveFrames. The meaning of this value is that the receiver is ready to receive a maximum of C_{TL_FCBS} (BlockSize) number of ConsecutiveFrames.
1_{16}	**Wait ($C_{TL\_FCFS(WAIT)}$)** The FlowControl Wait parameter shall be encoded by setting the lower nibble of the TL_PCI byte #1 to "1". It shall cause the sender to continue to wait for a new FlowControl TL_PDU and to restart its C_{TL_FCBS} timer. If FlowStatus is set to Wait, the values of C_{TL_FCBS} (BlockSize) and $C_{TL\_FCSTmin}$ (SeparationTime minimum) in the FlowControl message are not relevant and shall be ignored.
2_{16}	**Overflow ($C_{TL\_FCFS(OVFLW)}$)** The FlowControl Overflow parameter shall be encoded by setting the lower nibble of the TL_PCI byte #1 to "2". It shall cause the sender to abort the transmission of a segmented message and make a TL_Data.con service call with the parameter <TL_Result> = $C_{TL\_BUFFER\_OVFLW}$. This TL_PCI FlowStatus parameter value is only allowed to be transmitted in the FC TL_PDU that follows the FF TL_PDU and shall only be used if the message length FF_DL of the received FF TL_PDU exceeds the buffer size of the receiving entity. If FlowStatus is set to Overflow, the values of C_{TL_FCBS} (BlockSize) and $C_{TL\_FCSTmin}$ (SeparationTime minimum) in the FC TL_PDU are not relevant and shall be ignored.
3_{16} to F_{16}	**Reserved** This range of values is reserved by this document.

9.6.5.2 FlowStatus (FS) error handling

If an FC TL_PDU is received with an invalid (reserved) FS parameter value, the message transmission shall be aborted and the transport layer shall initiate a TL_Data.con service call with the parameter <TL_Result> = $C_{TL\_INVALID\_FS}$ (invalid FlowStatus) to the adjacent upper layer.

9.6.5.3 BlockSize (C_{TL_FCBS}) parameter definition

BlockSize (C_{TL_FCBS}): the maximum number of CF TL_PDUs the receiver allows the sender to send before waiting for an authorization to continue transmission of the following CF TL_PDUs. When C_{TL_FCBS} is set to zero by the receiver, the sender is not waiting for an authorization to continue the transmission.

The C_{TL_FCBS} parameter shall be encoded in byte #2 of the FC TL_PCI.

The units of C_{TL_FCBS} are the absolute number of CF TL_PDUs per block.

EXAMPLE If C_{TL_FCBS} is equal to 20, then the block consists of 20 CF TL_PDUs.

Only the last block of ConsecutiveFrames in a segmented data transmission may have less than the C_{TL_FCBS} number of CF TL_PDUs.

Table 20 provides an overview of the FC TL_PCI byte.

Table 20 — Definition of C_{TL_FCBS} values

Value	Description
00_{16}	**BlockSize (C_{TL_FCBS})** The C_{TL_FCBS} parameter value 0 shall be used to indicate to the sender that no more FC TL_PDUs shall be sent during the transmission of the segmented message. The sending transport layer entity shall send all remaining CF TL_PDUs without any stop for further FC TL_PDUs from the receiving transport layer entity.
01_{16} to FF_{16}	**BlockSize (C_{TL_FCBS})** This range of C_{TL_FCBS} parameter values shall be used to indicate to the sender the maximum number of CF TL_PDUs (ConsecutiveFrames) that can be received without an intermediate FC TL_PDU from the receiving transport layer entity.

9.6.5.4 SeparationTime minimum ($C_{TL\_FCSTmin}$) parameter definition

SeparationTime minimum ($C_{TL\_FCSTmin}$): the minimum time the sender is to wait between transmission of two CF TL_PDUs.

The $C_{TL\_FCSTmin}$ parameter shall be encoded in byte #3 of the FC TL_PDU (TL_PCI).

This time is specified by the receiving entity. The $C_{TL\_FCSTmin}$ parameter value specifies the minimum time gap allowed between the transmissions of two CF TL_PDUs (ConsecutiveFrame) transport protocol data units, see Table 21.

Table 21 — Definition of $C_{TL\_FCSTmin}$ values

Value	Description
00_{16} to $7F_{16}$	**SeparationTime minimum ($C_{TL\_FCSTmin}$) range: 0 ms to 127 ms** The units of $C_{TL\_FCSTmin}$ in the range 00_{16} to $7F_{16}$ (0 to 127) are absolute milliseconds [ms].
80_{16} to $F0_{16}$	**Reserved** This range of values is reserved by this document.
$F1_{16}$ to $F9_{16}$	**SeparationTime minimum ($C_{TL\_FCSTmin}$) range: 100 µs to 900 µs** The units of $C_{TL\_FCSTmin}$ in the range $F1_{16}$ to $F9_{16}$ are even multiples of 100 µs, where parameter value $F1_{16}$ represents 100 µs and parameter value $F9_{16}$ represents 900 µs.
FA_{16} to FF_{16}	**Reserved** This range of values is reserved by this document.

The measurement of the $C_{TL\_FCSTmin}$ starts after completion of transmission of a CF TL_PDU (ConsecutiveFrame) and ends at the request for the transmission of the next CF TL_PDU.

EXAMPLE If $C_{TL\_FCSTmin}$ is equal to 10 ($0A_{16}$), then the minimum SeparationTime authorized between CF TL_PDUs equals 10 ms.

The FlowControl mechanism (see Figure 9) allows the receiver to inform the sender about the receiver's capabilities, which the sender shall conform to.

As the values for C_{TL_FCBS} and $C_{TL\_FCSTmin}$ are provided by every received FC TL_PDU, two different modes for the adoption of these values are available for the receiver of a segmented message:

— dynamic: C_{TL_FCBS} and $C_{TL\_FCSTmin}$ are updated for the subsequent TL_PDU communication for this message;

— static: constant C_{TL_FCBS} and $C_{TL\_FCSTmin}$ values are used for the communication for this message.

All blocks, except the last one, consist of CF TL_PDUs. The last one contains the remaining CF TL_PDUs (from 1 up to C_{TL_FCBS}).

Each time the sender/receiver waits for a TL_PDU from the receiver/sender, a timeout mechanism allows detection of a transmission failure (see 9.8.2).

By means of FC TL_PDUs, the receiver has the possibility of authorizing transmission of the following CF TL_PDUs to delay transmission of that authorization or to deny reception of a segmented message in the case that the number of bytes to be transferred exceeds the number of bytes that can be stored in the receiver buffer:

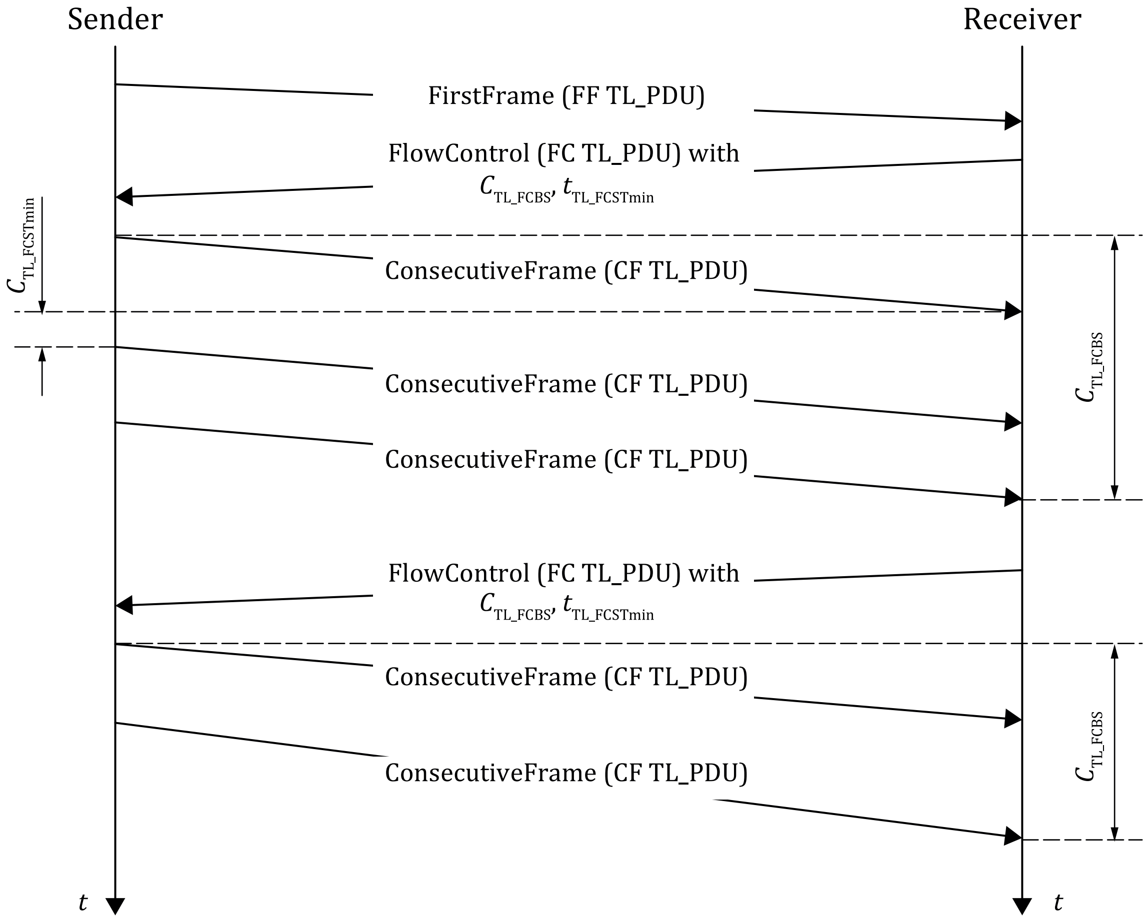
a) $C_{\text{TL\_FCFS(CTS)}}$: flow control flow status "ContinueToSend", the authorization to continue;

b) $C_{\text{TL\_FCFS(WAIT)}}$: flow control flow status request to continue to "WAIT";

c) $C_{\text{TL\_FCFS(OVFLW)}}$: flow control flow status buffer overflow, the indication that the number of bytes specified in the FF TL_PDU of the segmented message exceeds the number of bytes that can be stored in the buffer of the receiver entity.

There is an upper limit to the number of $C_{\text{TL\_FCFS(WAIT)}}$ a receiver is allowed to send in a row, called $C_{\text{TL\_FC\_WFTmax}}$. This parameter is a system design constant and is not transmitted in the FC TL_PDU.

Figure 9 shows the segmentation on the sender side and reassembly on the receiver side with FC TL_PDUs.



Key

t time

Figure 9 — FlowControl (FC) mechanism

9.6.5.5 SeparationTime minimum ($C_{\text{TL\_FCSTmin}}$) error handling

If an FC TL_PDU message is received with a reserved $C_{\text{TL\_FCSTmin}}$ parameter value, then the sending transport layer entity shall use the longest $C_{\text{TL\_FCSTmin}}$ value specified by this document ($7F_{16}$ = 127 ms) instead of the value received from the receiving transport layer entity for the duration of the ongoing segmented message transmission.

If the time between two subsequent CF TL_PDUs of a segmented data transmission ($t_{TL\_As} + t_{TL\_Cs}$) is smaller than the value commanded by the receiver via $C_{TL\_FCSTmin}$, there is no guarantee that the receiver of the segmented data transmission correctly receives and processes all TL_PDUs. In any case, the receiver of the segmented data transmission is not required to monitor adherence to the $C_{TL\_FCSTmin}$ value.

9.6.5.6 Dynamic C_{TL_FCBS}/$C_{TL\_FCSTmin}$ values in subsequent FC TL_PDUs

If the server is the receiver of a segmented message transfer (i.e. the sender of the FC TL_PDU), it may choose either to use the same values for C_{TL_FCBS} and $C_{TL\_FCSTmin}$ in subsequent FC TL_PDUs with $C_{TL\_FCFS(CTS)}$ of the same segmented message or to vary these values from FC TL_PDU to FC TL_PDU.

If the client, connected to an ISO 15765-compliant in-vehicle diagnostic network, is the receiver of a segmented message transfer (i.e. the sender of the FC TL_PDU), it shall use the same values for C_{TL_FCBS} and $C_{TL\_FCSTmin}$ in subsequent FC TL_PDUs with $C_{TL\_FCFS(CTS)}$ of the same segmented message.

If the client is the sender of a segmented data transmission (i.e. the receiver of the FC TL_PDU), it shall adjust to the values of C_{TL_FCBS} and $C_{TL\_FCSTmin}$ from each FC TL_PDUs with $C_{TL\_FCFS(CTS)}$ received during the same segmented data transmission.

NOTE For in-vehicle gateway (i.e. routing takes place on OSI layer 4; see ISO 14229-2), the vehicle manufacturer chooses either that the FC TL_PDU parameters C_{TL_FCBS} and $C_{TL\_FCSTmin}$ vary during the transmission of a single segmented message or that these parameters are static values. Depending on this design decision, the vehicle manufacturer ensures that the server is compatible with the respective in-vehicle gateway.

9.7 Maximum number of FC.WAIT frame transmissions (C_{TL_WFTmax})

The purpose of this variable is to avoid communication sender nodes being potentially hooked up in case of a fault condition whereby the latter could be waiting continuously. This parameter is local to communication peers and is not transmitted and is hence not part of the FC TL_PDU.

— The C_{TL_WFTmax} parameter shall indicate how many FC TL_PDU with $C_{TL\_FCFS(WAIT)}$ can be transmitted by the receiver in a row.

— The C_{TL_WFTmax} parameter upper limit shall be user defined at system generation time.

— The C_{TL_WFTmax} parameter shall only be used on the receiving transport layer entity during message reception.

— If the C_{TL_WFTmax} parameter value is set to zero, then FC TL_PDU shall rely upon $C_{TL\_FCFS(CTS)}$ (ContinueToSend) only. FC TL_PDU with $C_{TL\_FCFS(WAIT)}$ (WAIT) shall not be used by that transport layer entity.

9.8 Transport layer timing

9.8.1 Timing parameters

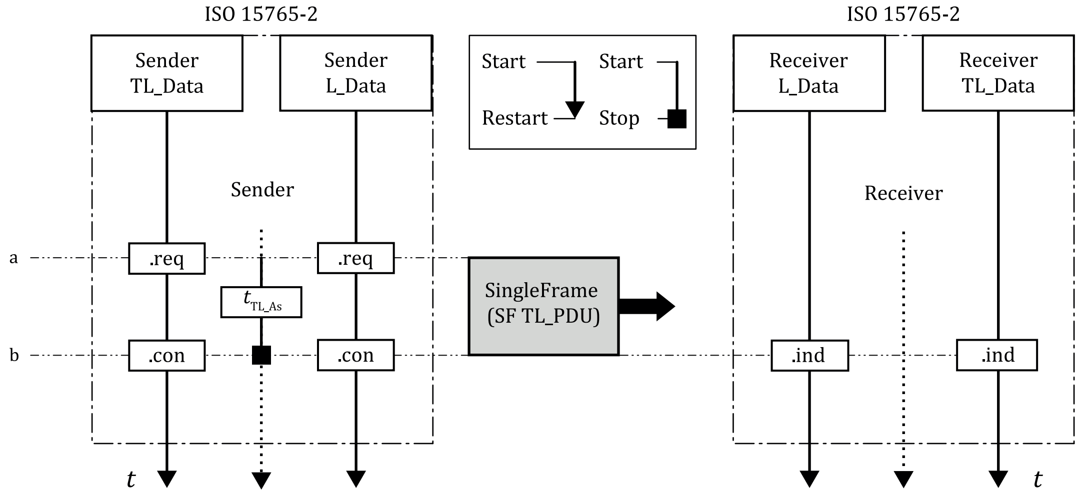
Performance requirement values are the binding communication requirements to be met by each communication peer in order to be compliant with this document. A certain application may use specific performance requirements within the ranges specified in Table 22.

Timeout values are specified to be higher than the values for the performance requirements within the ranges specified in Table 22 in order to ensure a working system and to overcome communication conditions, where the performance requirement can absolutely not be met (e.g. high bus load). Specified timeout values shall be treated as the lower limit. The real timeout shall occur no later than the specified timeout value +50 %.

The transport layer shall issue an appropriate service primitive to the network layer service user upon detection of an error condition.

If a communication path is established between peer protocols entities, identified by NL_AI (see 8.3.2.1 and 10.1.2 for further details), a single set of transport layer timing parameters is assigned statically to this communication path. For the selection of the transport layer timing parameters, no other information besides NL_AI is used. If different transport layer timing parameters are required for different use cases, then separate communication paths shall be established using different NL_AI parameters, e.g. different NL_TA and/or NL_SA shall be defined for each individual use case that requires different transport layer timing parameters.

Figure 10 shows the transport layer timing parameters of an unsegmented message while Table 22 specifies the transport layer timing parameter values and their corresponding start and end positions based on the data link layer services.



Key

t time

a Sender TL_Data.req: the session layer issues a message to the transport/network layer.

Sender L_Data.req: the transport/network layer transmits the SingleFrame TL_PDU to the data link layer and starts the t_{TL_As} timer.

b Receiver L_Data.ind: the data link layer issues to the transport/network layer the reception of the CAN frame.

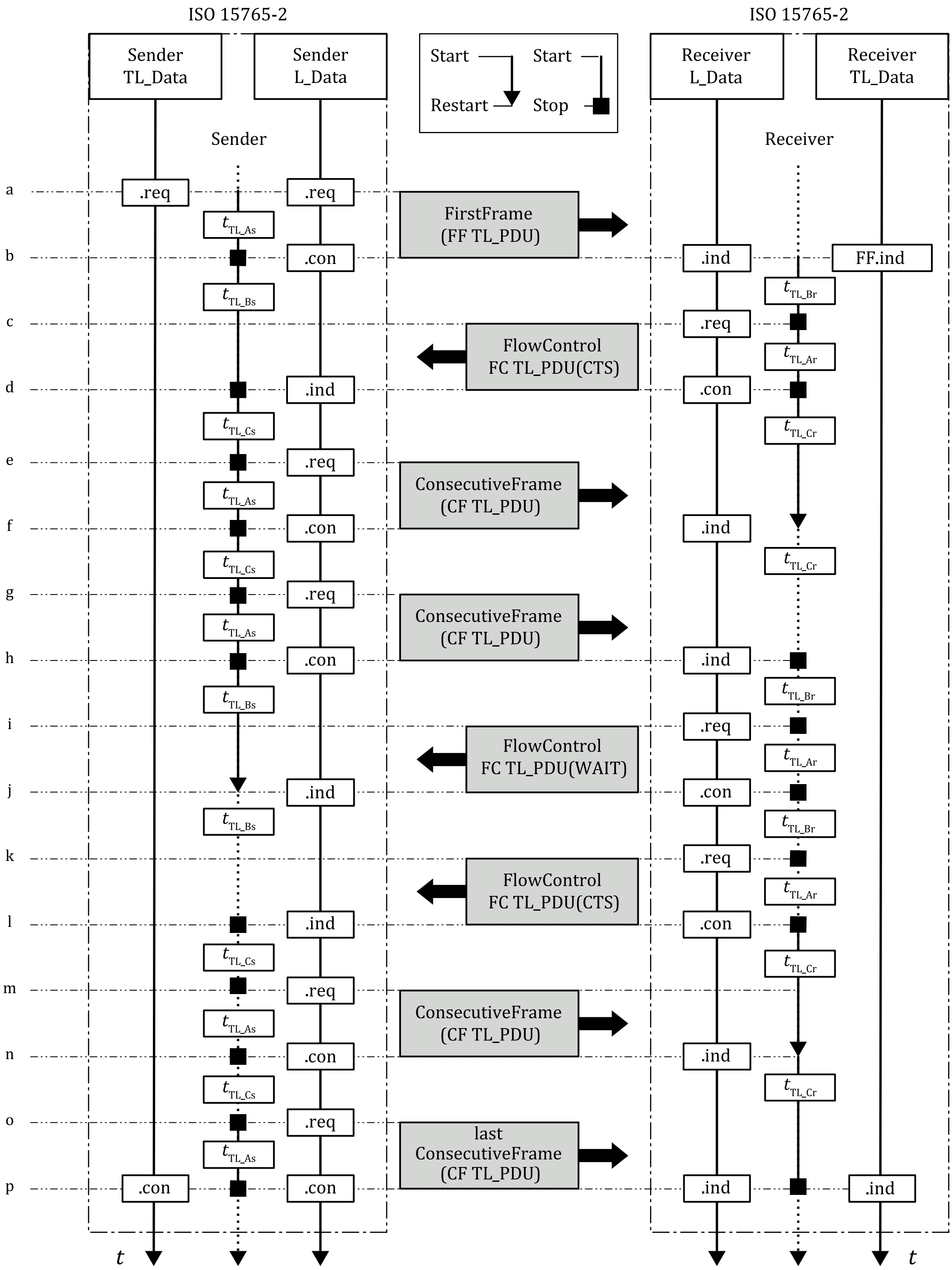
Receiver TL_Data.ind: the transport/network layer issues to the session layer the completion of the SingleFrame TL_PDU.

Sender L_Data.con: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the sender stops the t_{TL_As} timer.

Sender TL_Data.con: the transport/network layer issues to the session layer the completion of the SingleFrame TL_PDU.

Figure 10 — Placement of transport layer timing parameters — Unsegmented message

Figure 11 shows the transport layer timing parameters of a segmented message.



Key

t time

a `Sender TL_Data.req`: the session layer issues a message to the transport/network layer.

`Sender L_Data.req`: the transport/network layer transmits the FirstFrame TL_PDU to the data link layer and starts the t_{TL_As} timer.

b `Receiver L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the receiver starts the t_{TL_Br} timer.

`Receiver TL_DataFF.ind`: the transport/network layer issues to the session layer the reception of a FirstFrame TL_PDU of a segmented message.

`Sender L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the sender stops the t_{TL_As} timer and starts the t_{TL_Bs} timer.

c `Receiver L_Data.req`: the transport/network layer transmits the FlowControl TL_PDU (ContinueToSend and BlockSize value = 2_d) to the data link layer and starts the t_{TL_Ar} timer.

d `Sender L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the sender stops the t_{TL_Bs} timer and starts the t_{TL_Cs} timer.

`Receiver L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the receiver stops the t_{TL_Ar} timer and starts the t_{TL_Cr} timer.

e `Sender L_Data.req`: the transport/network layer transmits the first ConsecutiveFrame TL_PDU to the data link layer and starts the t_{TL_As} timer.

f `Receiver L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the receiver restarts the t_{TL_Cr} timer.

`Sender L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the sender stops the t_{TL_As} timer and starts the t_{TL_Cs} timer according to the separation time value ($C_{TL\_FCSTmin}$) of the previous FlowControl TL_PDU.

g `Sender L_Data.req`: when the t_{TL_Cs} timer is elapsed ($C_{TL\_FCSTmin}$), the transport/network layer transmits the next ConsecutiveFrame TL_PDU to the data link layer and starts the t_{TL_As} timer.

h `Receiver L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the receiver stops the t_{TL_Cr} timer and starts the t_{TL_Br} timer.

`Sender L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the sender stops the t_{TL_As} timer and starts the t_{TL_Bs} timer; the sender is waiting for the next FlowControl TL_PDU.

i `Receiver L_Data.req`: the transport/network layer transmits the FlowControl TL_PDU (Wait) to the data link layer and starts the t_{TL_Ar} timer.

j `Sender L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the sender restarts the t_{TL_Bs} timer.

`Receiver L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the receiver stops the t_{TL_Ar} timer and starts the t_{TL_Br} timer.

k `Receiver L_Data.req`: the transport/network layer transmits the FlowControl TL_PDU (ContinueToSend) to the data link layer and starts the t_{TL_Ar} timer.

l `Sender L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the sender stops the t_{TL_Bs} timer and starts the t_{TL_Cs} timer.

`Receiver L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the receiver stops the t_{TL_Ar} timer and starts the t_{TL_Cr} timer.

m `Sender L_Data.req`: the transport/network layer transmits the ConsecutiveFrame TL_PDU to the data link layer and starts the t_{TL_As} timer.

n `Receiver L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the receiver restarts the t_{TL_Cr} timer.

`Sender L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the sender stops the t_{TL_As} timer and starts the t_{TL_Cs} timer according to the separation time value ($C_{TL\_FCSTmin}$) of the previous FlowControl TL_PDU.

o `Sender L_Data.req`: when the t_{TL_Cs} timer is elapsed ($C_{TL\_FCSTmin}$), the transport/network layer transmits the last ConsecutiveFrame TL_PDU to the data link layer and starts the t_{TL_As} timer.

p `Receiver L_Data.ind`: the data link layer issues to the transport/network layer the reception of the CAN frame; the receiver stops the t_{TL_Cr} timer.

`Receiver TL_Data.ind`: the transport/network layer issues to the session layer the completion of the segmented message.

`Sender L_Data.con`: the data link layer confirms to the transport/network layer that the CAN frame has been acknowledged; the sender stops the t_{TL_As} timer.

`Sender TL_Data.con`: the transport/network layer issues to session layer the completion of the segmented message.

Figure 11 — Placement of transport layer timing parameters — Segmented message

Table 22 specifies the transport layer timing parameter values and their corresponding start and end positions based on the data link layer services.

Table 22 — Transport layer timing parameter values

Timing parameter	Description	Data link layer service Start	End	Timeout ms	Performance ms
t_{TL_As}	Time for transmission of the CAN frame (any TL_PDU) on the sender side	`L_Data.req`	`L_Data.con`	1 000	—
t_{TL_Ar}	Time for transmission of the CAN frame (any TL_PDU) on the receiver side	`L_Data.req`	`L_Data.con`	1 000	—
t_{TL_Bs}	Time until reception of the next FlowControl TL_PDU	`L_Data.con` (FF) `L_Data.con` (CF) `L_Data.ind` (FC)	`L_Data.ind` (FC TL_PDU)	1 000	—
t_{TL_Br}	Time until transmission of the next FlowControl TL_PDU	`L_Data.ind` (FF) `L_Data.ind` (CF) `L_Data.con` (FC)	`L_Data.req` (FC TL_PDU)	N/A	$(t_{TL\_Br} + t_{TL\_Ar})$ < (0,9 × t_{TL_Bs} timeout)
t_{TL_Cs}	Time until transmission of the next ConsecutiveFrame TL_PDU	`L_Data.ind` (FC) `L_Data.con` (CF)	`L_Data.req` (CF TL_PDU)	N/A	$(t_{TL\_Cs} + t_{TL\_As})$ < (0,9 × t_{TL_Cr} timeout)
t_{TL_Cr}	Time until reception of the next Consecutive Frame TL_PDU	`L_Data.con` (FC) `L_Data.ind` (CF)	`L_Data.ind` (CF TL_PDU)	1 000	—

9.8.2 Transport layer timeouts

Table 23 specifies the cause and action in a transport layer timeout.

Table 23 — Transport layer timeout error handling

Error	Cause	Action
t_{TL_As}	Any TL_PDU not transmitted in time on the sender side	Abort message transmission and issue `TL_Data.con` with `<TL_Result>` = $C_{TL\_TIMEOUT\_As}$
t_{TL_Ar}	Any TL_PDU not transmitted in time on the receiver side	Abort message reception and issue `TL_Data.ind` with `<TL_Result>` = $C_{TL\_TIMEOUT\_Ar}$
t_{TL_Bs}	FlowControl TL_PDU not received (lost, overwritten) on the sender side or preceding FirstFrame TL_PDU or ConsecutiveFrame TL_PDU not received (lost, overwritten) on the receiver side	Abort message transmission and issue `TL_Data.con` with `<TL_Result>` = $C_{TL\_TIMEOUT\_Bs}$
t_{TL_Cr}	ConsecutiveFrame TL_PDU not received (lost, overwritten) on the receiver side or preceding FC TL_PDU not received (lost, overwritten) on the sender side	Abort message reception and issue `TL_Data.ind` with `<TL_Result>` = $C_{TL\_TIMEOUT\_Cr}$

9.8.3 Unexpected arrival of TL_PDU

An unexpected TL_PDU is defined as one that has been received by a node outside the expected order of TL_PDUs. It could be a TL_PDU defined by this document (SF TL_PDU, FF TL_PDU, CF TL_PDU or FC TL_PDU) that is received out of the normal expected order or else it could be an unknown TL_PDU that cannot be interpreted by the definitions given in this document.

As a general rule, arrival of an unexpected TL_PDU from any node shall be ignored, with the exception of SF TL_PDUs and physically addressed FF TL_PDUs; functionally addressed FF TL_PDUs shall be ignored. When the specified action is to ignore an unexpected TL_PDU, this means that the transport layer shall not notify the upper layers of its arrival.

Depending on the transport layer design decision to support full- or half-duplex communication, the interpretation of "unexpected" differs:

a) with half-duplex, point-to-point communication between two nodes is only possible in one direction at a time;

b) with full-duplex, point-to-point communication between two nodes is possible in both directions at once.

In addition to this network layer design decision, it is necessary to consider the possibility that a reception or transmission from/to a node with the same address information (NL_AI) as contained in the received unexpected NL_PDU is in progress.

Table 24 specifies the transport layer behaviour in the case of the reception of an unexpected TL_PDU, in consideration of the actual transport layer internal status and the design decision to support half- or full-duplex communication. Table 24 only applies if the received NL_PDU contains the same NL_AI as the reception or transmission that is in progress at the time the TL_PDU is received.

If the NL_AI of the received NL_PDU is different from the expected received NL_AI segmented message, an on-going reception/transmission shall be continued.

For a segmented transmission or reception, the same NL_AI as currently being processed means the same NL_AI in the received NL_PDUs associated with the message.

Table 24 — Handling of unexpected arrival of TL_PDU with same NL_AI as currently being processed

NWL status	Reception of			FC TL_PDU	Unknown TL_PDU
	SF TL_PDU	FF TL_PDU	CF TL_PDU		
Segmented transmit in progress	Full-duplex: if a reception is in progress, refer to table cell [“SF TL_PDU” | “Segmented receive in progress”]; otherwise, process the SF TL_PDU as the start of a new reception.	Full-duplex: if a reception is in progress, refer to table cell [“FF TL_PDU” | “Segmented receive in progress”]; otherwise, process the FF TL_PDU as the start of a new reception.	Full-duplex: if a reception is in progress, refer to table cell [“CF TL_PDU” | “Segmented receive in progress”].	Ignore[a]	Ignore
	Half-duplex: ignore	Half-duplex: ignore	Half-duplex: ignore		
Segmented receive in progress	Terminate the current reception, report a `TL_Data.ind`, with <TL_Result> set to TL_UNEXP_PDU, to the upper layer, and process the SF TL_PDU as the start of a new reception.	Terminate the current reception, report a `TL_Data.ind`, with <TL_Result> set to TL_UNEXP_PDU, to the upper layer, and process the FF TL_PDU as the start of a new reception.	Ignore[b]: if awaited, process the CF TL_PDU in the on-going reception and perform the required checks (e.g. C_{TL_CFSN} in right order); otherwise, ignore it.	Ignore	Ignore
Idle[c]	Process the SF TL_PDU as the start of a new reception.	Process the FF TL_PDU as the start of a new reception.	Ignore	Ignore	Ignore

[a] FC TL_PDU parameter error handling is described separately in 9.6.5.2 and 9.6.5.5.

[b] Handling of an unexpected SN is described separately in 9.6.4.4.

[c] Neither a segmented transmission nor a segmented reception is in progress. This status “Idle” only describes the transport layer itself and is not an indication of the availability of the layers above, which might be busy and thus might not be able to accept a new (SF TL_PDU) request or provide a diagnostic buffer for the data of a multi-frame request (FF TL_PDU).

9.8.4 Wait frame error handling

If the receiver has transmitted C_{TL_WFTmax} in an FC TL_PDU(WAIT) in a row and, following this, it cannot meet the performance requirement for the transmission of a $C_{TL\_FCFS(CTS)}$ FC TL_PDU(CTS), then the receiver side shall abort the message reception and issue a `TL_Data.ind` with <TL_Result> set to $C_{TL\_WFT\_OVRN}$ to the higher layer.

The sender of the message is informed about the aborted message reception via a `TL_Data.con` with <TL_Result> set to $t_{TL\_TIMEOUT\_Bs}$ (because of the missing FC TL_PDU from the receiver, a $t_{TL\_TIMEOUT\_Bs}$ timeout occurs in the sender).

9.9 Interleaving of messages

The network layer protocol shall be capable of carrying out parallel transmission of different messages that are not mapped onto the same NL_AI. This is necessary to ensure that the receiving peer is able to reassemble the received network protocol data units in a consistent manner. This scheme enables, for example, gateway operation that needs to handle different message transmissions concurrently across distinct sub-networks.

10 Network layer protocol

10.1 Protocol data unit field specification

10.1.1 NL_PDU format

The protocol data unit (NL_Data) enables the transfer of data between the network layer in one node and the network layer in one or more other nodes (peer protocol entities). All NL_PDUs consist of three fields, as given in Table 25.

Table 25 — NL_PDU format

Address information	Protocol control information	Data field
NL_AI	TL_PCI	A_PDU

10.1.2 Address information (NL_AI)

The NL_AI is used to identify the communicating peer entities of the network layer. The NL_AI information received in the NL_SDU (NL_SA, NL_TA, NL_TAtype [and NL_AE]) shall be copied and included in the NL_PDU.

The NL_AI shall be included (repeated) in every NL_PDU that is transmitted.

The NL_AI contains address information that identifies the type of message exchanged and the recipient(s) and sender between whom data exchange takes place.

NOTE For a detailed description of address information, see 8.3.2.

10.2 Creating CAN frames based on NL_TAtype and TX_DL

CAN frames are generated based upon NL_AI (see 8.3.2), the configured addressing format for the given NL_AI, the configured TX_DL value and the size of the message to be transmitted.

10.3 Mapping of the NL_PDU fields

10.3.1 Addressing formats

The exchange of network layer data is supported by three addressing formats: normal, extended and mixed addressing. Each addressing format requires a different number of CAN frame data bytes to encapsulate the addressing information associated with the data to be exchanged. Consequently, the number of data bytes transported within a single CAN frame depends on the type of addressing format chosen.

8.3.2.1 to 8.3.2.5 specify the mapping mechanisms for each addressing format based on the data link layer services and service parameters specified in ISO 11898-1.

10.3.2 Normal addressing

For 11-bit CAN identifiers, the mapping of the NL_AI target address (TA) and source address (SA) into a CAN identifier is implied.

Table 26 specifies the mapping of NL_PDU parameters into CAN frame where the addressing format is normal and NL_TAtype indicates the message is physical.

Table 26 — Mapping of NL_PDU parameters into CAN frame — Normal addressing, NL_TAtype = #1, #3, #5 and #7

NL_PDU type	CAN ID	Data field Byte 1 to n[a]
SingleFrame (SF)	NL_AI	TL_PCI, A_PDU
FirstFrame (FF)	NL_AI	TL_PCI, A_PDU
ConsecutiveFrame (CF)	NL_AI	TL_PCI, A_PDU
FlowControl (FC)	NL_AI	TL_PCI, flow status
[a] See Table 2 and Table 10.		

Table 27 specifies the mapping of NL_PDU parameters into CAN frame where the addressing format is normal and NL_TAtype indicates the message is functional.

Table 27 — Mapping of NL_PDU parameters into CAN frame — Normal addressing, NL_TAtype = #2, #4, #6 and #8

NL_PDU type	CAN ID	Data field Byte 1 to n[a]
SingleFrame (SF)	NL_AI	TL_PCI, A_PDU
[a] See Table 3 and Table 9.		

10.3.3 Normal fixed addressing

Normal fixed addressing is a subformat of normal addressing in which the mapping of the address information into the CAN identifier is further defined. In the general case of normal addressing, described above, the correspondence between NL_AI and the CAN identifier is left open.

For normal fixed addressing, only 29-bit CAN identifiers are allowed. Table 28 and Table 29 specify the mapping of the address information (NL_AI) into the CAN identifier, depending on the target address type (NL_TAtype). TL_PCI and A_PDU are placed in the CAN frame data field.

Table 28 specifies normal fixed addressing where NL_TAtype indicates the message is physical.

Table 28 — Normal fixed addressing, NL_TAtype = #5 and #7

NL_PDU type	29-bit CAN ID bit position						Data field
	28 – 26	25	24	23 – 16	15 – 8	7 – 0	Byte 1 to n[a]
SingleFrame (SF)	110_2	0	0	218	NL_TA	NL_SA	TL_PCI, A_PDU
FirstFrame (FF)	110_2	0	0	218	NL_TA	NL_SA	TL_PCI, A_PDU
ConsecutiveFrame (CF)	110_2	0	0	218	NL_TA	NL_SA	TL_PCI, A_PDU
FlowControl (FC)	110_2	0	0	218	NL_TA	NL_SA	TL_PCI, flow status
[a] See Table 2 and Table 10.							

Table 29 specifies normal fixed addressing where NL_TAtype indicates the message is functional.

Table 29 — Normal fixed addressing, NL_TAtype = #6 and #8

NL_PDU type	29-bit CAN ID bit position						Data field
	28 – 26	25	24	23 – 16	15 – 8	7 – 0	Byte 1 – n[a]
SingleFrame (SF)	110_2	0	0	219	NL_TA	NL_SA	TL_PCI, A_PDU
[a] See Table 2 and Table 10.							

A.2.1 specifies the normal fixed addressing with data link layer according to SAE J1939 and shall be followed.

10.3.4 Extended addressing

For 11-bit CAN identifiers, the mapping of the NL_AI (except NL_TA) source address (SA) into a CAN identifier is implied.

Table 30 specifies the mapping of NL_PDU parameters into CAN frame where the addressing format is extended and NL_TAtype indicates the message is physical.

Table 30 — Mapping of NL_PDU parameters into CAN frame — Extended addressing, NL_TAtype = #1, #3, #5 and #7

NL_PDU type	CAN ID	Data field	
		Byte 1	Byte 2 to n[a]
SingleFrame (SF)	NL_AI, except NL_TA	NL_TA	TL_PCI, A_PDU
FirstFrame (FF)	NL_AI, except NL_TA	NL_TA	TL_PCI, A_PDU
ConsecutiveFrame (CF)	NL_AI, except NL_TA	NL_TA	TL_PCI, A_PDU
FlowControl (FC)	NL_AI, except NL_TA	NL_TA	TL_PCI, flow status

[a] See Table 2 and Table 10.

Table 31 specifies the mapping of NL_PDU parameters into CAN frame where the addressing format is extended and NL_TAtype indicates the message is functional.

Table 31 — Mapping of NL_PDU parameters into CAN frame — Extended addressing, NL_TAtype = #2, #4, #6 and #8

NL_PDU type	CAN ID	Data field	
		Byte 1	Byte 2 to n[a]
SingleFrame (SF)	NL_AI, except NL_TA	NL_TA	TL_PCI, A_PDU

[a] See Table 2 and Table 10.

10.3.5 Mixed addressing

10.3.5.1 29-bit CAN identifier

Mixed addressing is the addressing format to be used if Mtype is set to remote diagnostics.

Table 32 and Table 33 specify the mapping of the address information (NL_AI) into the 29-bit CAN identifier scheme and the first CAN frame data byte, depending on the target address type (NL_TAtype). TL_PCI and A_PDU are placed in the remaining bytes of the CAN frame data field.

Table 32 — Mixed addressing with 29-bit CAN identifier, NL_TAtype = #5 and #7

NL_PDU type	29-bit CAN ID bit position						Data field	
	28 – 26	25	24	23 – 16	15 – 8	7 – 0	Byte 1	Byte 2 to n[a]
SingleFrame (SF)	110_2	0	0	206	NL_TA	NL_SA	NL_AE	TL_PCI, A_PDU
FirstFrame (FF)	110_2	0	0	206	NL_TA	NL_SA	NL_AE	TL_PCI, A_PDU
ConsecutiveFrame (CF)	110_2	0	0	206	NL_TA	NL_SA	NL_AE	TL_PCI, A_PDU
FlowControl (FC)	110_2	0	0	206	NL_TA	NL_SA	NL_AE	TL_PCI, flow status

[a] See Table 2 and Table 10.

Table 33 — Mixed addressing with 29-bit CAN identifier, NL_TAtype = #6 and #8

NL_PDU type	29-bit CAN ID bit position						Data field	
	28 – 26	25	24	23 – 16	15 – 8	7 – 0	Byte 1	Byte 2 to n[a]
SingleFrame (SF)	110_2	0	0	205	NL_TA	NL_SA	NL_AE	TL_PCI, A_PDU
[a] See Table 2 and Table 10.								

A.2.2 specifies the mixed addressing with data link layer according to SAE J1939.

10.3.5.2 11-bit CAN identifier

Mixed addressing is the addressing format to be used if Mtype is set to remote diagnostics.

Table 34 and Table 35 specify the mapping of the address information (NL_AI) into the 11-bit CAN identifier scheme. For 11-bit CAN identifiers, the mapping of the NL_AI target address (NL_TA) and source address (NL_SA) into a CAN identifier is implied. For each combination of NL_SA, NL_TA and NL_TAtype, the same CAN identifier can be used. NL_AE is placed in the first data byte of the CAN frame data field. TL_PCI and A_PDU are placed in the remaining bytes of the CAN frame data field.

Table 34 — Mixed addressing with 11-bit CAN identifier, NL_TAtype = #1 and #3

NL_PDU type	CAN ID	Data field	
		Byte 1	Byte 2 to n[a]
SingleFrame (SF)	NL_AI	NL_AE	TL_PCI, A_PDU
FirstFrame (FF)	NL_AI	NL_AE	TL_PCI, A_PDU
ConsecutiveFrame (CF)	NL_AI	NL_AE	TL_PCI, A_PDU
FlowControl (FC)	NL_AI	NL_AE	TL_PCI, flow status
[a] See Table 2 and Table 10.			

Table 35 — Mixed addressing with 11-bit CAN identifier, NL_TAtype = #2 and #4

NL_PDU type	CAN ID	Data field	
		Byte 1	Byte 2 to n[a]
SingleFrame (SF)	NL_AI	NL_AE	TL_PCI, A_PDU
[a] See Table 2 and Table 10.			

11 Data link layer usage

11.1 Data link layer service parameters

The following data link layer service parameters are specified in ISO 11898-1:

— `<Data>`: CAN frame data;

— `<DLC>`: data length code;

— `<Identifier>`: CAN identifier; CAN identifier values shall conform to the restrictions as specified in Annex B;

— `<Transfer_Status>`: status of a transmission;

— `<Format>`: frame format (CAN, CAN FD, base: 11-bit, extended: 29-bit) (see Table 2).

11.2 Data link layer interface services

The data link layer abstract service primitive interface `L_Data.req`, `L_Data.ind`, and `L_Data.con` is specified in ISO 11898-1.

11.3 CAN frame data length code (DLC)

11.3.1 DLC parameter

The DLC parameter specifies the number of data bytes transmitted in a CAN frame. This document does not specify any requirements concerning the length of the data field in a CAN frame other than those implied by the size of the network layer protocol data units.

An application that implements the network layer as specified in this document might either pad all CAN frames to their full length (see 11.3.2.1) or optimize the DLC to the applicable length of the L_PDU (see 11.3.2.2). CAN frames according to ISO 11898-1 (CAN FD frame type) can require a mandatory padding for DLC values greater than 8 (see 11.3.2.3).

11.3.2 CAN frame data

11.3.2.1 Data field padding (TX_DL = 8)

If data field padding is used, the DLC is always set to 8, even if the L_PDU to be transmitted is shorter than 8 bytes. The sender shall pad any unused bytes in the frame, e.g. as depicted in Table 36. In particular, this can be the case for an SF TL_PDU, FC TL_PDU or the last CF TL_PDU of a segmented message. If not specified differently, the default value CC_{16} should be used for L_PDU padding in order to minimize the stuff-bit insertions and bit alterations on the wire.

The DLC parameter of the CAN frame is set by the sender and read by the receiver to determine the number of data bytes per CAN frame to be processed by the data link layer. The DLC parameter cannot be used to determine the message length; this information shall be extracted from the TL_PCI information at the beginning of a message.

Table 36 — Data padding example (TX_DL = 8), normal addressing, L_PDU size 6 byte, DLC = 8

L_PDU type	CAN ID	Data field (Byte)							
		1	2	3	4	5	6	7	8
Data link layer frame	NL_AI	TL_PCI	A_PDU					padding	
	345_{16}	05_{16}	44_{16}	55_{16}	66_{16}	77_{16}	88_{16}	CC_{16}	CC_{16}

11.3.2.2 Data field optimization (TX_DL = 8)

If data field optimization is used, the DLC does not always need to be 8. If the L_PDU to be transmitted is shorter than 8 bytes, then the sender can optimize the CAN bus load by shortening the data field to contain only the number of bytes occupied by the L_PDU (no padding of unused data bytes). Data field optimization can only be used for an SF TL_PDU, FC TL_PDU or the last CF TL_PDU of a segmented message.

EXAMPLE See Table 37.

The DLC parameter of the CAN frame is set by the sender and read by the receiver to determine the number of data bytes per CAN frame to be processed by the data link layer. The DLC parameter cannot be used to determine the message length; this information shall be extracted from the TL_PCI information in the beginning of a message.

Table 37 — Data optimized example (TX_DL = 8), normal addressing, L_PDU size 6 byte, DLC = 6

L_PDU type	CAN ID	Data field (byte)					
		1	2	3	4	5	6
Data link layer frame	NL_AI	TL_PCI	A_PDU				
	345_{16}	05_{16}	44_{16}	55_{16}	66_{16}	77_{16}	88_{16}

11.3.2.3 Mandatory padding of CAN FD frames (TX_DL > 8)

According to ISO 11898-1, the data length code (DLC) from 0 to 8 specifies the CAN frame payload length in byte (1:1 mapping). For L_PDU length, values up to 8 bytes either the data padding (see 11.3.2.1) or the DLC data optimization (see 11.3.2.2) are applicable.

The ISO 11898-1 DLC values from 9 to 15 are assigned to nonlinear discrete values for CAN frame payload length up to 64 byte. To prevent the transmission of uninitialized data, the padding of CAN frame data is mandatory for DLC values greater than 8 when the length of the L_PDU size to be transmitted is not equal to one of the discrete length values specified in ISO 11898-1:—, Table 5 (DLC). An example is shown in Table 38. For DLC values from 9 to 15, only the mandatory padding shall be used. If not specified differently, the value CC_{16} should be used for padding as default, to minimize the stuff-bit insertions and bit alterations on the wire.

The DLC parameter of the CAN frame is set by the sender and read by the receiver to determine the number of data bytes per CAN frame to be processed by the data link layer. The DLC parameter cannot be used to determine the message length; this information shall be extracted from the TL_PCI information in the beginning of a message.

Table 38 — Data padding example (TX_DL > 8), normal addressing, L_PDU size 11 bytes, DLC = 9

L_PDU type	CAN ID	Data field (Byte)											
		1	2	3	4	5	6	7	8	9	10	11	12
Data link layer frame	NL_AI	TL_PCI		A_PDU									padding
	345_{16}	00_{16}	09_{16}	11_{16}	22_{16}	33_{16}	44_{16}	55_{16}	66_{16}	77_{16}	88_{16}	99_{16}	CC_{16}

NOTE ISO 11898-1:—, Table 5 (DLC) value 9 leads to a CAN FD frame payload length of 12 byte.

11.3.3 Data length code (DLC) error handling

Depending on the TL_PCI value, the data link layer can calculate the smallest expected value for the CAN DLC parameter in a received CAN frame.

Reception of a CAN frame with a DLC value smaller than expected (less than 8 for applications which pad the CAN frames or smaller than implied by the size of the data link layer L_PDU for data optimization) shall be ignored by the data link layer without any further action.

Annex A
(normative)

Use of normal fixed and mixed addressing according to SAE J1939-21

A.1 Overview

This annex describes how to map address information parameters, NL_AI, into the CAN frame when a data link layer according to SAE J1939 is used.

A.2 Rules

A.2.1 Normal fixed addressing

Table A.1 shows the mapping of address information parameters, NL_AI, into the CAN frame when network target address type, NL_TAtype, physical addressing is used.

Table A.1 — Normal addressing, physical addressed messages

SAE J1939 name	P	R	DP	PF	PS	SA	Data field
Bits	3	1	1	8	8	8	64
Content	default 110_2	0	0	218	NL_TA	NL_SA	TL_PCI, A_PDU
CAN ID bits	28 to 26	25	24	23 to 16	15 to 8	7 to 0	—
CAN data byte	—	—	—	—	—	—	1 to 8
CAN field	Identifier						Data
NOTE See A.2.3 to A.2.8 for definitions of the abbreviated terms used in this table.							

Table A.2 shows the mapping of address information parameters, NL_AI, into the CAN frame when network target address type, NL_TAtype, functional addressing is used.

Table A.2 — Normal addressing, functional addressed messages

SAE J1939 name	P	R	DP	PF	PS	SA	Data field
Bits	3	1	1	8	8	8	64
Content	default 110_2	0	0	219	NL_TA	NL_SA	TL_PCI, A_PDU
CAN ID bits	28 to 26	25	24	23 to 16	15 to 8	7 to 0	—
CAN data byte	—	—	—	—	—	—	1 to 8
CAN field	Identifier						Data
NOTE See A.2.3 to A.2.8 for definitions of the abbreviated terms used in this table.							

A.2.2 Mixed addressing

Table A.3 shows the mapping of address information parameters, NL_AI, into the CAN frame when the network target address type, NL_TAtype, physical addressing is used.

Table A.3 — Mixed addressing, physical addressed messages

SAE J1939 name	P	R	DP	PF	PS	SA	Data field	
Bits	3	1	1	8	8	8	8	56
Content	default 110_2	0	0	206	NL_TA	NL_SA	N_AE	TL_PCI, A_PDU
CAN ID bits	28 to 26	25	24	23 to 16	15 to 8	7 to 0		—
CAN data byte	—	—	—	—	—	—	1	2 to 8
CAN field	Identifier						Data	
NOTE See A.2.3 to A.2.8 for definitions of the abbreviated terms used in this table.								

Table A.4 shows the mapping of address information parameters, NL_AI, into the CAN frame when network target address type, NL_TAtype, functional addressing is used.

Table A.4 — Mixed addressing, functional addressed messages

SAE J1939 name	P	R	DP	PF	PS	SA	Data field	
Bits	3	1	1	8	8	8	8	56
Content	default 110_2	0	0	205	NL_TA	NL_SA	N_AE	TL_PCI, A_PDU
CAN ID bits	28 to 26	25	24	23 to 16	15 to 8	7 to 0		—
CAN data byte	—	—	—	—	—	—	1	2 to 8
CAN field	Identifier						Data	
NOTE See A.2.3 to A.2.8 for definitions of the abbreviated terms used in this table.								

A.2.3 Priority (P)

The priority is user defined with a default value of six.

The 3-bit priority field is used to optimize message latency for transmission onto the CAN bus only. The priority field should be masked off by the receiver (ignored). The priority of any CAN message can be set from highest, 0 (000 bin.), to lowest, 7 (111 bin.).

A.2.4 Reserved bit (R)

The reserved bit shall be set to “0”.

A.2.5 Data page (DP)

The data page bit shall be set to “0”.

A.2.6 Protocol data unit format (PF)

The format is of the type PDU1, “destination specific”.

Diagnostic messages shall use the following parameter group numbers (PGN):

— mixed addressing: 52736 for NL_TAtype = #5, which gives PF = 206;

— mixed addressing: 52480 for NL_TAtype = #6, which gives PF = 205;

— normal fixed addressing: 55808 for NL_TAtype = #5, which gives PF = 218;

— normal fixed addressing: 56064 for NL_TAtype = #6, which gives PF = 219.

A.2.7 PDU-specific (PS)

The PDU-specific field shall contain the target address (destination address), NL_TA.

A.2.8 Source address (SA)

The SA field shall contain the source address, NL_SA.

A.2.9 Update rate

Update rate is specified according to user requirements.

A.2.10 Data length

Data length shall be 8 bytes.

Annex B
(normative)

Reserved CAN IDs

The purpose of this annex is to reserve CAN IDs out of the 11-bit CAN ID range for future use in International Standards.

Table B.1 specifies the reserved CAN ID ranges.

Table B.1 — ISO-reserved CAN IDs

CAN identifier	Description
$7F4_{16}$ to $7F6_{16}$	Reserved by this document
$7FA_{16}$ to $7FB_{16}$	Reserved by this document

Bibliography

[1] ISO/IEC 10731, *Information technology — Open Systems Interconnection — Basic Reference Model — Conventions for the definition of OSI services*

[2] ISO 14229-2, *Road vehicles — Unified diagnostic services (UDS) — Part 2: Session layer services*

[3] ISO 15031 (all parts), *Road vehicles — Communication between vehicle and external equipment for emissions-related diagnostics*

[4] ISO 15765-4, *Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 4: Requirements for emissions-related systems*

[5] ISO 15765-5, *Road vehicles — Diagnostic communication over Controller Area Network (DoCAN) — Part 5: Specification for an in-vehicle network connected to the diagnostic link connector*

[6] ISO 26021 (all parts), *Road vehicles — End-of-life activation of in-vehicle pyrotechnic devices*

[7] ISO 27145 (all parts), *Road vehicles — Implementation of World-Wide Harmonized On-Board Diagnostics (WWH-OBD) communication requirements*

[8] SAE J1939-21, *Data Link Layer*

[9] SAE J1979, *E/E Diagnostic Test Modes*

This page is intentionally blank.

This page is intentionally blank.

ICS 43.040.15; 43.180

Price based on 47 pages

iso.org