INTERNATIONAL STANDARD

ISO 10681-2

First edition
2010-06-15

Road vehicles — Communication on FlexRay —

Part 2: Communication layer services

Véhicules routiers — Communication par FlexRay —

Partie 2: Services de la couche de communication

Reference number
ISO 10681-2:2010(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 10681-2 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 10681 consists of the following parts, under the general title *Road vehicles — Communication on FlexRay*:

— *Part 1: General information and use case definition*

— *Part 2: Communication layer services*

Introduction

This part of ISO 10681 is based on the Open Systems Interconnection (OSI) Basic Reference Model specified in ISO/IEC 7498-1 and ISO/IEC 10731, which structures communication systems into seven layers (see example in Table 1). When mapped on this model, this part of ISO 10681 incorporates the network layer (layer 3) and the transport layer (layer 4) services as communication layer services.

Table 1 — Example of enhanced diagnostic specifications according to the OSI layers

Applicability	OSI layers	Vehicle manufacturer enhanced diagnostics
Seven layers according to ISO/IEC 7498-1 and ISO/IEC 10731	Application layer	ISO 14229-1
	Presentation layer	N/A
	Session layer	ISO 14229-2
	Transport layer	ISO 10681-2
	Network layer	
	Data link layer	FlexRay Communications Systems Protocol Specification
	Physical layer	FlexRay Communications System Electrical Physical Layer Specification

Road vehicles — Communication on FlexRay —

Part 2: Communication layer services

1 Scope

This part of ISO 10681 specifies the requirements for a communication protocol tailored to meet the requirements of FlexRay-based vehicle network systems as specified in the FlexRay Communications Systems Protocol Specification. As the communication protocol combines the network layer and transport layer functionality (OSI layers 3 and 4), this part of ISO 10681 does not explicitly distinguish between these layer services.

The technical features of this communication protocol are as follows:

— transmit messages with known data length;

— transmit messages with unknown but finite data length;

— additional acknowledgement with retry mechanism;

— routing data on the fly;

— support of dynamic frame length.

2 Normative references

The following referenced documents are indispensable for the application 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 7498-2, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 2: Security Architecture*

ISO/IEC 7498-3, *Information technology — Open Systems Interconnection — Basic Reference Model: Naming and addressing*

ISO/IEC 7498-4, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 4: Management framework*

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

3 Terms, definitions, symbols and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 7498-1, ISO 7498-2, ISO/IEC 7498-3 and ISO/IEC 7498-4 and the following apply.

3.1.1
communication layer
CL
layer that includes the network layer (layer 3) and the transport layer (layer 4)

3.1.2
protocol data unit
PDU
⟨layered system⟩ data unit that is specified in the protocol of a given layer

NOTE The protocol data unit contains user data of that layer and possible protocol control information. The protocol data unit of layer X is the service data unit of its lower layer (X – 1).

3.1.3
service data unit
SDU
⟨layered system⟩ set of data that is sent by a user of the service in a given layer

NOTE It is transmitted to a peer service user with no semantic change.

3.2 Abbreviated terms

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

ABT	abort
ACK	acknowledge
BC	bandwidth control
BfS	buffer size
BP	byte position
C_AI	communication address information
C_Ar	communication layer timing parameter A receiver
C_As	communication layer timing parameter A sender
C_Br	communication layer timing parameter B receiver
C_Bs	communication layer timing parameter B sender
C_Cr	communication layer timing parameter C receiver
C_Cs	communication layer timing parameter C sender
C_CT	communication type
C_Data	communication layer data transfer service name
C_PCI	communication protocol control information
C_SA	communication source address
C_TA	communication target address

C_TAType	communication target address type
CF	consecutive frame
CL	communication layer
COMM	communication
CTS	continue to send
C_PDU	communication layer protocol data unit
C_SDU	communication layer service data unit
DLL	data link layer
ECU	electronic control unit
EOB	end of block
FC	flow control
FIFO	first in first out
FPL	frame payload length
FR	FlexRay
FS	flow status
Ind	indication
LF	last frame
L_PDU	data link layer protocol data unit
max	maximum
ML	message length
MNPC	maximum numbers of PDUs per cycle
N/A	not applicable
N_PDU	network protocol data unit
NUM	number
OVFLW	overflow
PDU	protocol data unit
pLatestTx	latest point of transmission
Req	request
RET	retry
RX	receive
SCexp	separation cycle exponent
SDU	service data unit
SN	sequence number
STF	start frame
STFA	start frame acknowledged
STFU	start frame unacknowledged

UNEXP	unexpected
UNUM	unsigned numeric
TX	transmission
WFT	wait frame transmission
WT	wait

3.3 Symbols

Σ	summation
$\neq$	not equal

4 Conventions

ISO 10681 is based on the conventions discussed in the OSI Services Conventions (ISO/IEC 10731:1994) as they apply for diagnostic services.

5 Communication layer overview

5.1 General

This clause describes the overall functionality of the communication layer. This part of ISO 10681 specifies an unacknowledged and acknowledged communication layer protocol for the exchange of data with known or unknown length between network nodes, e.g. from ECU to ECU, or between a diagnostic tester equipment and an ECU. If the data to be transferred do not fit into a single L_PDU, a segmentation method is provided.

In order to describe the function of the communication layer, services provided to upper layers and the internal operation of the communication layer have to be considered.

5.2 Services provided by communication layer to upper layers

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

Four types of services are defined.

a) Communication services

These services, of which the following are defined, enable the transfer of up to 64 kbytes of data.

1) C_Data.request

This service is used to request the transfer of data. If necessary, the communication layer segments the data.

2) C_Data_STF.indication

This service is used to signal the beginning of a segmented message reception to the upper layer.

3) C_Data.indication

This service is used to provide received data to the upper layer.

4) C_Data.confirm

This service confirms to the upper layer 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) C_ChangeParameter.request

This service is used to request the dynamic setting of specific internal parameters.

2) C_ChangeParameter.confirm

This service confirms to the upper layer that the request to change a specific parameter has been carried out (successfully or not).

3) C_GetParameter.request

This service is used to request the value of a communication layer parameter for a given connection.

4) C_GetParameter.confirm

This service is used to return the value of a communication layer parameter for a given connection.

c) Status services

1) C_GetStatus.request

This service is used to request the status of a transfer of data (transmit/receive).

2) C_GetStatus.confirm

This service confirms to the upper layer the status of the transfer of data (transmit/receive).

d) Transmission control services

C_Cancel.request

This service is used to request cancellation of an ongoing message transmission. This request can only be issued by the sender of a message, after the message transmission has been started via a C_Data.request service primitive.

5.3 Internal operation of communication layer

5.3.1 General

The internal operation of the communication layer provides the following methods for segmentation, transmission with flow control, and reassembly:

— transmission of a message with a known message length;

— transmission of a message with an unknown but finite message length;

— acknowledgement of the transmission with a retry mechanism.

The main purpose of the communication layer is to perform a transfer of a message that might or might not fit in a single FlexRay frame. Messages which do not fit into a single FlexRay frame are segmented into multiple parts, where each can be transmitted in a FlexRay frame.

5.3.2 Rules

The communication layer establishes certain rules for an ongoing transmission/reception, which are briefly described here for better understanding of the examples given in 5.3.3. For details on certain frames and their usage, see 7.4.

— Segmented messages are always concluded with a LastFrame. This frame might include the last bytes of the message if they did not fit completely into the previous ConsecutiveFrame.

— An acknowledged message transmission that is started with a StartFrame_ACK is always concluded by the receiver with a FlowControl_ACK after the reception of the last frame of the message, which can either be a single StartFrame or a LastFrame (in the case of a multiframe transmission).

— The request to acknowledge the reception of a block of ConsecutiveFrames via the ConsecutiveFrame_EOB shall always be confirmed by the receiver with a FlowControl frame which indicates the status of the reception (CTS, WAIT, etc.).

— In the case of an unknown message length transmission, the upper layer of the sender either asynchronously or synchronously provides data during the ongoing message transmission via the C_Data.request service primitive.

— In the case of a known message length transmission where the amount of data fits into a single STF and is completely available for transmission, a LastFrame will not be transmitted.

— Functional requests are transmitted as unsegmented and unacknowledged messages. Any other format will be ignored by the communication layer (no error will be communicated).

— If an STFU payload length is greater than the possible payload in an STFU, then the frame shall be ignored by the receiver.

5.3.3 Message sequence charts

All examples given assume a communication between the sender and the receiver on a single subnet, not taking into account communication over gateways.

Figure 1 shows an example of an unsegmented unacknowledged message transmission with a known message length.

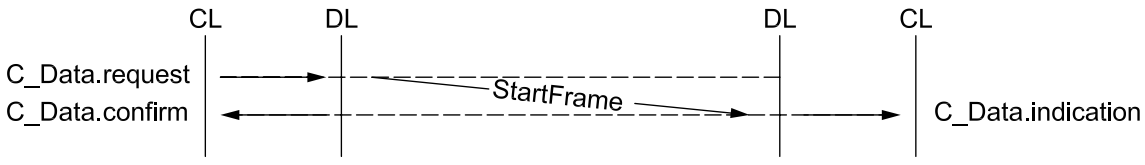


Figure 1 — Example of unsegmented unacknowledged message (known message length)

Figure 2 shows an example of an unsegmented acknowledged message transmission with a known message length.

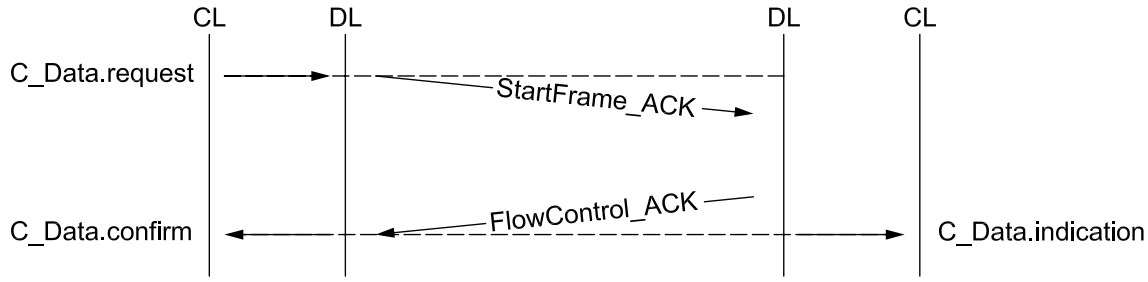


Figure 2 — Example of unsegmented acknowledged message (known message length)

Figure 3 shows an example of a segmented unacknowledged message transmission with a known message length.

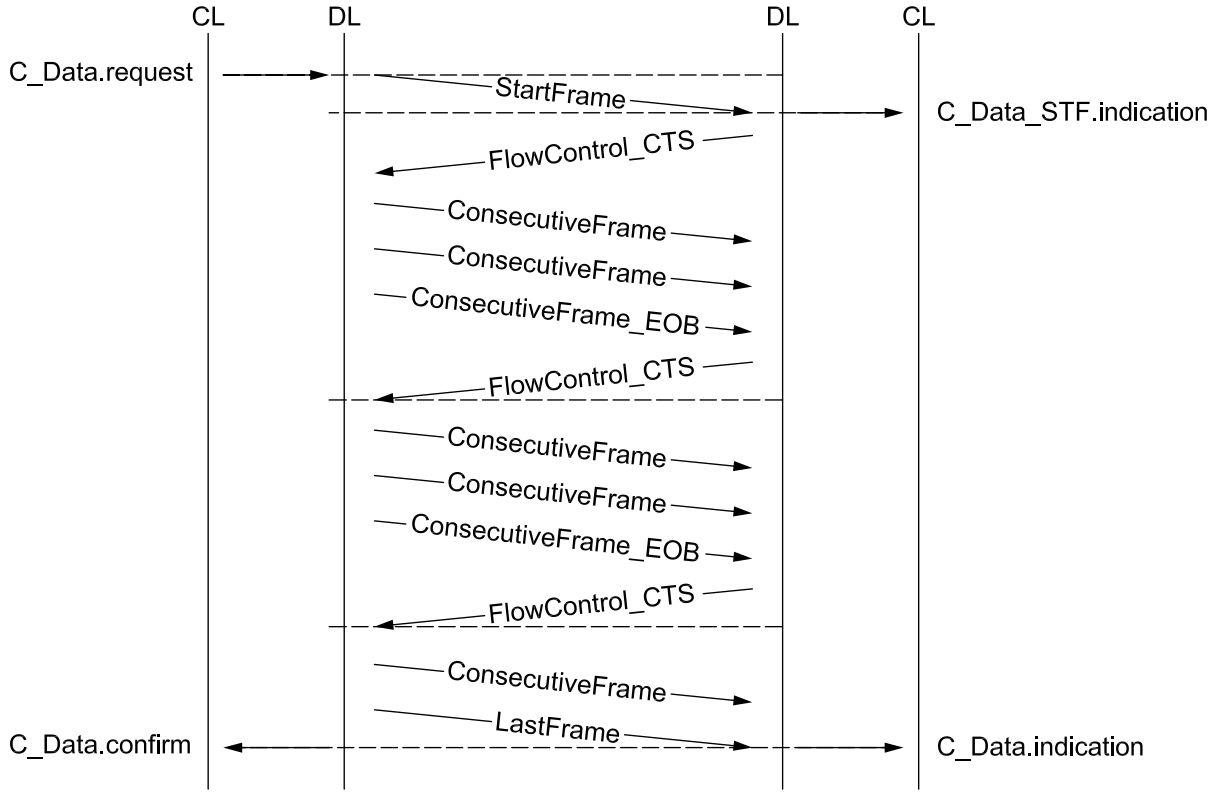


Figure 3 — Example of segmented unacknowledged message (known message length)

Figure 4 shows an example of a segmented acknowledged message transmission with a known message length.

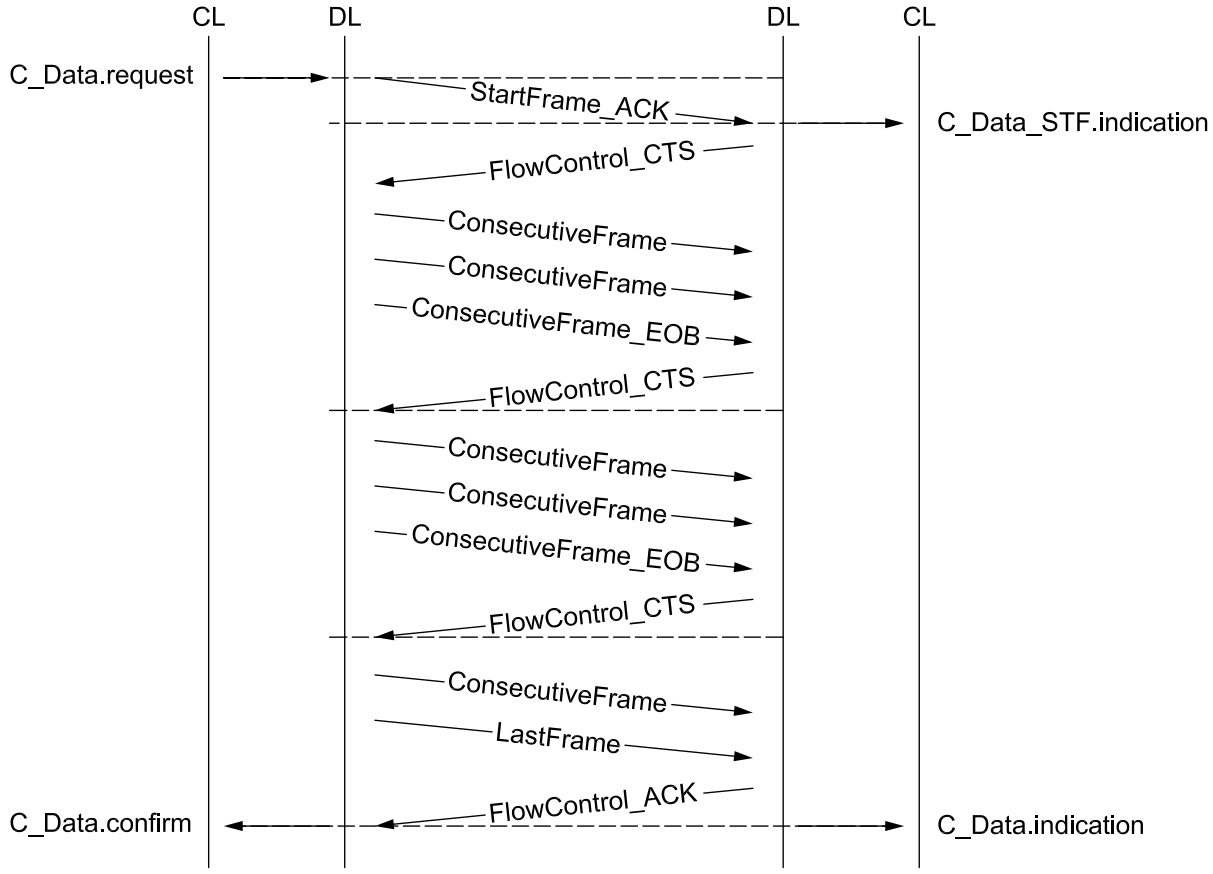


Figure 4 — Example of segmented acknowledged message (known message length)

Figure 5 shows an example of a segmented unacknowledged message transmission with an unknown message length.

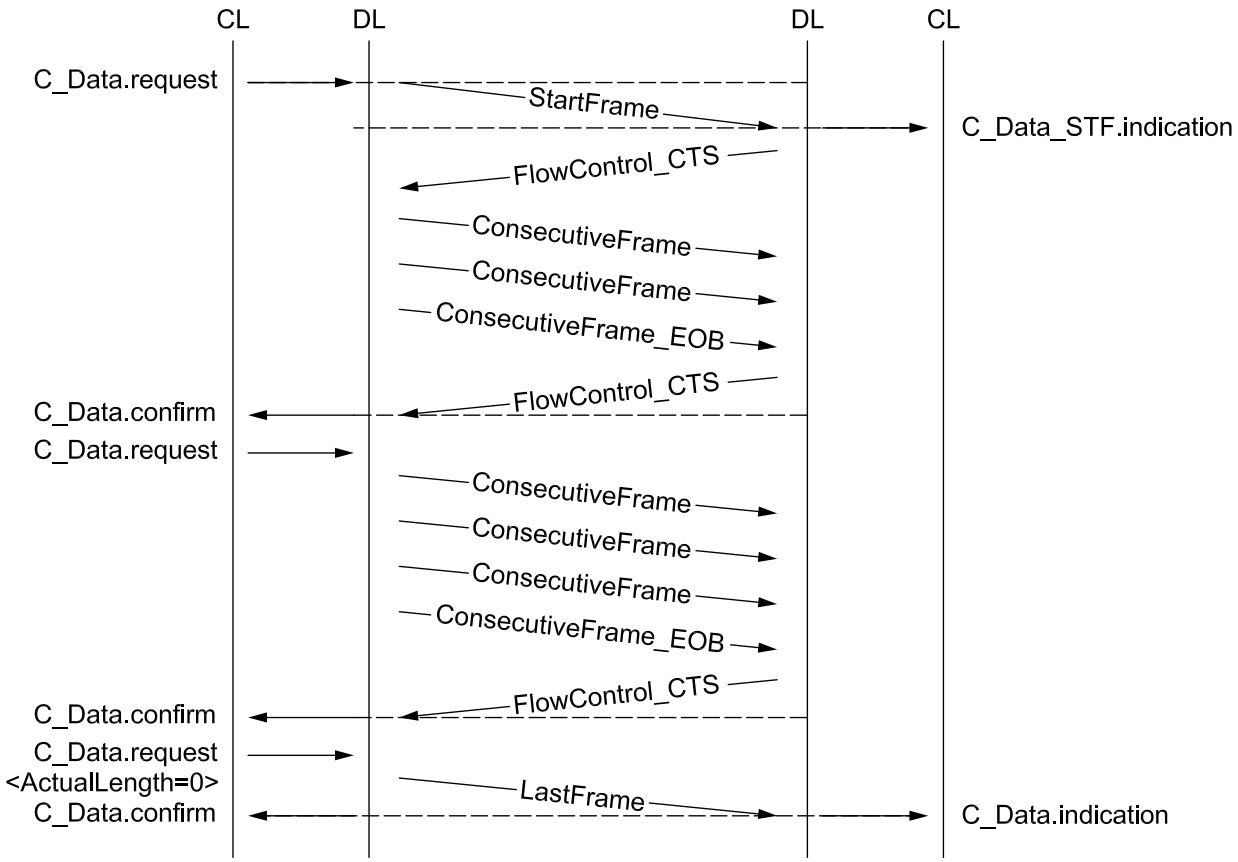


Figure 5 — Example of segmented unacknowledged message (unknown message length)

Figure 6 shows an example of a segmented acknowledged message transmission with an unknown message length.

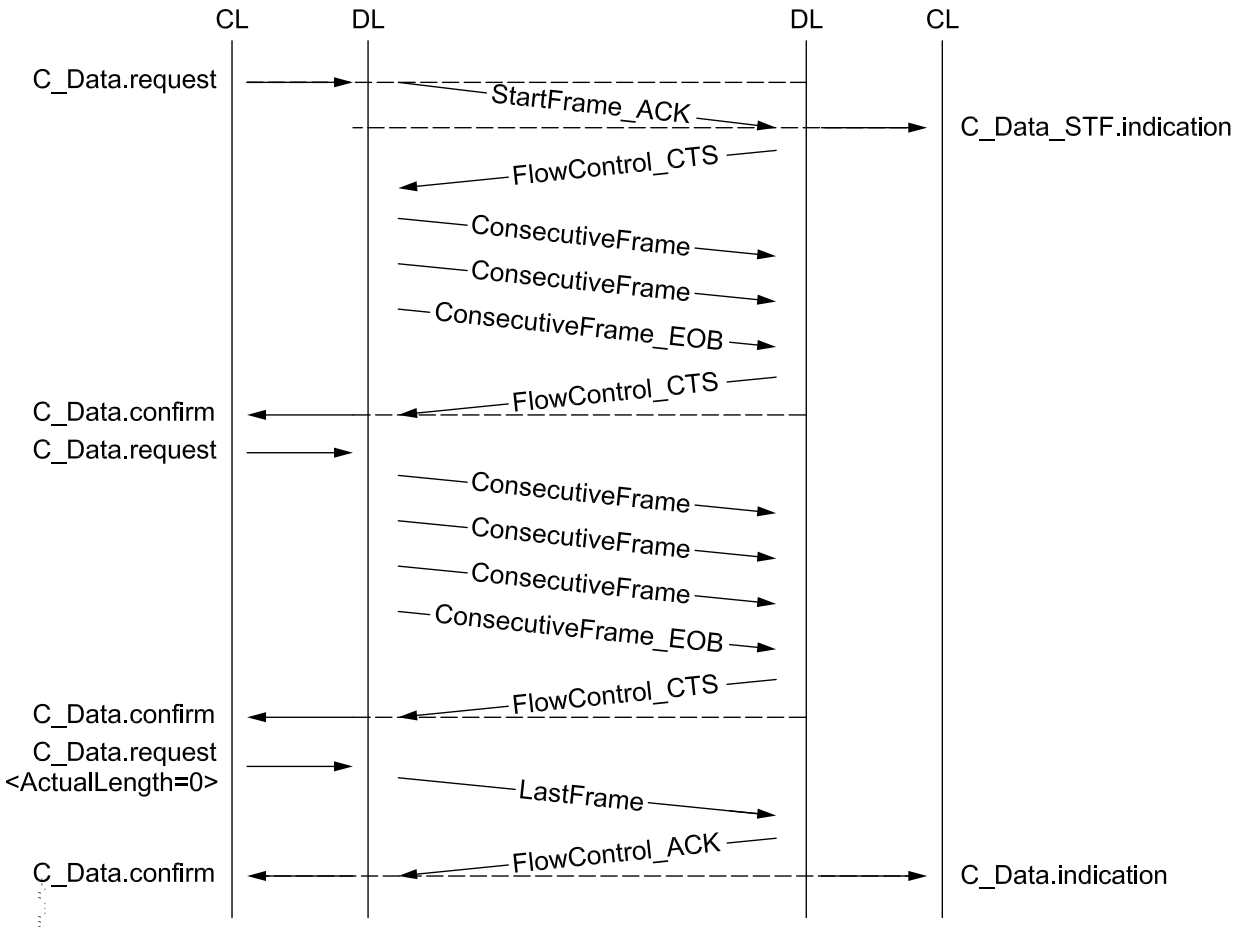


Figure 6 — Example of segmented acknowledged message (unknown message length)

6 Communication layer services

6.1 General

All communication layer services have the same general structure. To define the services, three types of service primitives are specified:

— a service *request* primitive, used by upper layers to pass control information and the data required to be transmitted to the communication layer;

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

— a service *confirmation* primitive used by the communication layer to pass status information to upper layers.

This service specification only specifies a set of service primitives that are independent of any implementation[1]. It does not specify an application programming interface.

All communication 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. For C_Data, “type” indicates the type of the service primitive, and “parameter A, parameter B, parameter C, ...” are the C_SDU presented as a list of values passed by the service primitive.

6.2 Specification of communication layer service primitives

6.2.1 C_Data.request

The service primitive requests transmission of <MessageData> with <Length> number of bytes from the sender to the receiver peer entities identified by the address information in C_SA, C_TA, C_TAtype (see 6.3 for parameter definitions). Furthermore, <C_CT> is supported in order to distinguish between an acknowledged and an unacknowledged data transfer. For the support of an unknown message length transmission, an indication of the <ActualLength> number of bytes is provided.

Each time the C_Data.request service is called, the communication layer shall signal the completion (or failure) of the message transmission to the service user by means of the issuing of a C_Data.confirm service call:

```
C_Data.request    (
                  C_SA,
                  C_TA,
                  C_TAtype,
                  C_CT,
                  <Length>,
                  <ActualLength>,
                  <MessageData>
                  )
```

6.2.2 C_Data.confirm

The C_Data.confirm service is issued by the communication layer. The service primitive confirms the completion of a C_Data.request service identified by the address information in C_SA, C_TA and C_TAtype. The parameter <C_Result> provides the status of the service request (see 6.3 for parameter definitions).

```
C_Data.confirm    (
                  C_SA,
                  C_TA,
                  C_TAtype,
                  <C_Result>
                  )
```

1) Annex A includes a detailed example on how the different scenarios of buffer resources in the sender and receiver are handled. Those examples make use of the service primitives as they would be API functions to describe the required behavior.

6.2.3 C_Data_STF.indication

The C_Data_STF.indication service is issued by the communication layer. The service primitive indicates to the upper layer the arrival of a start frame (STF) of a segmented message received from a peer protocol entity, identified by the address information in C_SA, C_TA and C_TAtype (see 6.3 for parameter definitions). This indication shall take place upon reception of the start frame (STF) of a segmented message.

C_Data_STF.indication (C_SA, C_TA, C_TAtype, <Length>)

The C_Data_STF.indication service shall always be followed by a C_Data.indication service call from the communication layer, indicating the completion (or failure) of the message reception.

A C_Data_STF.indication service call shall only be issued by the communication layer if a correct start frame (STF) message segment has been received.

If the communication layer detects an error in a start frame (STF), then the message is ignored by the communication layer and no C_Data_STF.indication shall be issued to the upper layer.

6.2.4 C_Data.indication

The C_Data.indication service is issued by the communication layer. The service primitive indicates <C_Result> events and delivers <MessageData> with <Length> bytes received from a peer protocol entity identified by the address information in C_SA, C_TA and C_TAtype to the upper layer (see 6.3 for parameter definitions).

The parameters <MessageData> and <Length> are only valid if <C_Result> equals C_OK.

C_Data.indication (C_SA, C_TA, C_TAtype, <C_Result> <Length>, <MessageData>)

The C_Data.indication service call is issued after the reception of a single start frame (STF) that includes the complete message (unsegmented message) or as an indication of the completion (or failure) of a segmented message reception.

If the communication layer detects any type of error in a single start frame (STF), then the message shall be ignored by the communication layer and no C_Data.indication shall be issued to the adjacent upper layer.

6.2.5 C_ChangeParameter.request

The service primitive is used to request the change of an internal parameter's value on the local protocol entity identified by the address information in C_SA, C_TA and C_TAtype. The <Parameter_Value> is assigned to the <Parameter> (see 6.3 for parameter definitions).

A parameter change is always possible, except after reception of the start frame (C_Data_STF.indication) and until the end of the reception of the corresponding message (C_Data.indication).

C_ChangeParameter.request (
C_SA,
C_TA,
C_TAtype,
<Parameter>,
<Parameter_Value>
)

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

6.2.6 C_ChangeParameter.confirm

The service primitive confirms the completion of a C_ChangeParameter.request service applying to a message identified by the address information in C_SA, C_TA and C_TAtype (see 6.3 for parameter definitions).

C_ChangeParameter.confirm (
C_SA,
C_TA,
C_TAtype
<Parameter>
<Result_ChangeParameter>
)

6.2.7 C_GetParameter.request

The service primitive is used to request the configured internal parameter's value on the local protocol entity identified by the address information in C_SA, C_TA and C_TAtype (see 6.3 for parameter definitions). A parameter read is always possible.

C_GetParameter.request (
C_SA,
C_TA,
C_TAtype
<Parameter>
)

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

6.2.8 C_GetParameter.confirm

The service primitive confirms the completion of a C_GetParameter.request service applying to a message identified by the address information in C_SA, C_TA and C_TAtype (see 6.3 for parameter definitions).

C_GetParameter.confirm (
C_SA,
C_TA,
C_TAtype
<Parameter>
<Result_GetParameter>
<Parameter_Value>
)

6.2.9 C_GetStatus.request

The service primitive requests status of an ongoing communication is identified by the address information in C_SA, C_TA and C_TAtype (see 6.3 for parameter definitions).

C_GetStatus.request (
C_SA,
C_TA,
C_TAtype
<StatusParameter>)

6.2.10 C_GetStatus.confirm

The C_GetStatus.confirm service is issued by the communication layer. The service primitive confirms the completion of a C_GetStatus.request service identified by the address information in C_SA, C_TA and C_TAtype. The parameter <Status_Value> provides the status of the ongoing communication (see 6.3 for parameter definitions).

C_GetStatus.confirm (
C_SA,
C_TA,
C_TAtype
<StatusParameter>
<Result_GetStatus>
<Status_Value>
)

6.2.11 C_Cancel.request

The service primitive requests cancellation of an ongoing transmission is identified by the address information in C_SA, C_TA and C_TAtype (see 6.3 for parameter definitions).

C_Cancel.request (
C_SA,
C_TA,
C_TAtype)

There is no confirmation service primitive defined. If a new transmission is issued via C_Data.request before the C_Cancel.request has been served completely, the upper layer will receive a C_Data.confirm with C_Result = C_TX_ON.

6.3 Service data unit specification

6.3.1 C_AI, Address Information

6.3.1.1 C_AI description

These parameters refer to addressing information. As a whole, the C_AI parameters are used to identify the source address (C_SA) and target address (C_TA) of message senders and recipients. See Table 2.

Table 2 — C_AI format

Byte 1	Byte 2	Byte 3	Byte 4
15	0	15	0
C_TA		C_SA	

6.3.1.2 C_SA, Communication Source Address

Type: 16 bit unsigned integer value

Range: 0000 to FFFF hex

Description: The C_SA parameter shall be used to encode the sending communication layer protocol entity.

NOTE The communication layer uses this parameter to identify corresponding C_PDUs, but neither modifies it nor defines values for the C_SA.

6.3.1.3 C_TA, Communication Target Address

Type: 16 bit unsigned integer value

Range: 0000 to FFFF hex

Description: The C_TA parameter shall be used to encode the receiving communication layer protocol entity.

NOTE The communication layer uses this parameter to identify corresponding C_PDUs, but does neither modify it nor define values for the C_TA.

6.3.1.4 C_TAtype, Communication Target Address type

Type: enumeration

Range: physical, functional

Description: The parameter C_TAtype is a configuration attribute to the C_TA parameter and implicitly part of C_TA. It shall be used to encode the communication model used by the communicating peer entities of the communication layer. Two communication models are specified: 1-to-1 communication, called “physical addressing” (unicast), and 1-to-n communication, called “functional addressing” (multicast/broadcast).

— Physical addressing (1-to-1 communication – unicast) shall be supported for all types of communication layer messages (unsegmented and segmented).

— Functional addressing (1-to-n communication – multicast/broadcast) shall only be supported for unsegmented, unacknowledged, known message length communication messages.

6.3.1.5 C_CT, Communication Type

Type: enumeration

Range: unacknowledged, acknowledged

Description: The C_CT parameter shall be used to specify the type of communication to be performed from a sender perspective. The parameter is implicitly configured within C_SA and C_TA.

6.3.2 <C_Result>

Type: enumeration

Range: C_OK, C_MOREDATA, C_TIMEOUT_A, C_TIMEOUT_Bs, C_TIMEOUT_Cr, C_WRONG_SN, C_ML_MISMATCH, C_INVALID_FS, C_UNEXP_PDU, C_WFT_OVRN, C_ABORT, C_CANCEL, C_WRONG_BP, C_BUFFER_OVFLW, C_ERROR

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 communication layer entity shall use the first parameter value found in this list in the error indication to the upper layers.

NOTE Table 3 implicitly defines the priority of the result value evaluation by the order of the result values (the first entry has the highest priority).

Table 3 — C_Result values

C_Result	Sender	Receiver	Description
C_TIMEOUT_A	X		This value is issued to the protocol user when the timer C_As has passed its time-out value C_As_{max}.
		X	This value is issued to the protocol user when the timer C_Ar has passed its time-out value C_Ar_{max}.
C_TIMEOUT_Bs	X		This value is issued to the service user when the timer C_Bs has passed its time-out value C_Bs_{max}.
C_TIMEOUT_Cr		X	This value is issued to the service user when the timer C_Cr has passed its time-out value C_Cr_{max}.
C_ML_MISMATCH		X	This value indicates a mismatch of the message length information (ML) within the STF PDU, the sum of the FPL information and the LF C_PDU (check of the ML = 0 in the case an unknown message length is supported).
C_WRONG_SN		X	This value is issued to the service user upon reception of an unexpected sequence number (PCI.SN) value. For more details, see 7.5.7.2 *SN Error handling*.
C_INVALID_FS	X		This value is issued to the service user when an invalid or unknown FlowStatus value has been received in a flow control (FC) C_PDU.
C_UNEXP_PDU		X	This value is issued to the service user upon reception of an unexpected protocol data unit.
C_WFT_OVRN		X	The receiver shall wait until the transmission of the (WFTmax + 1) FC__WT C_PDU would occur, but does not transmit this FC C_PDU and instead indicates the error C_WFT_OVRN to the upper layer. The sender detects a C_Bs timeout and terminates the transmission.
C_ABORT	X		This value is issued to the service user upon reception of a flow control (FC) C_PDU with FlowStatus = ABT. It indicates that the receiver is currently busy and cannot take the request at that point in time; therefore, the transmission of the segmented message was aborted.
C_CANCEL	X		This value is issued to the service user upon the cancellation of an ongoing transmission as requested by the upper application via the cancel service primitive.
C_BUFFER_OVFLW	X		This value is issued to the service user upon reception of a flow control (FC) C_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 StartFrame MessageLength (STF_ML) parameter in the StartFrame; therefore, the transmission of the segmented message was aborted.
C_WRONG_BP	X		This value is issued to the service user upon reception of a flow control (FC) C_PDU with FlowStatus = ACK_RET, where BP points to a position outside the buffer of the sender. Therefore, the sender cannot provide the requested data anymore. The segmented message was aborted.
C_ERROR	X	X	This is the general error value. It shall be issued to the service user when an error has been detected by the communication layer and no other parameter value can be used to better describe the error.

Table 3 (*continued*)

C_Result	Sender	Receiver	Description
C_MOREDATA	X		This value does not indicate an error. For the case of restricted buffer size in the *sender*, this value is issued to the service user in the sender upon the completion of a block transmission in order to inform the upper layer to feed in further data of the ongoing message to be transmitted.
		X	For the case of restricted buffer size in the *receiver*, this value is issued to the service user in the receiver upon the reception of a block of data in order to inform the upper layer that more data is expected to come.
C_TX_ON	X		This value indicates that the current request for a message transmission has not been accepted due to that situation that there is already a transmission for the specified address information <AI> active. The upper layer has to repeat the request for the transmission, because the communication layer does not buffer the request. This value can be issued to a service user on the sender side only.
C_OK	X	X	This value does not indicate an error; it means that the service execution has been completed successfully.

6.3.3 <Length>

Type: UNUM16

Range: 0000 to FFFF hex

Description: This parameter includes the length of data to be transmitted/received. A zero indicates an unknown message length transmission and <ActualLength> indicates the handed over number of data bytes.

6.3.4 <ActualLength>

Type: UNUM16

Range: 0000 to FFFF hex

Description: This parameter includes the length of data actually included in the request primitive in the case <Length> is zero.

Figure 7 shows how to interpret the values of the parameters <ActualLength> and <Length> in order to perform the transmission of a message of either known or unknown length.

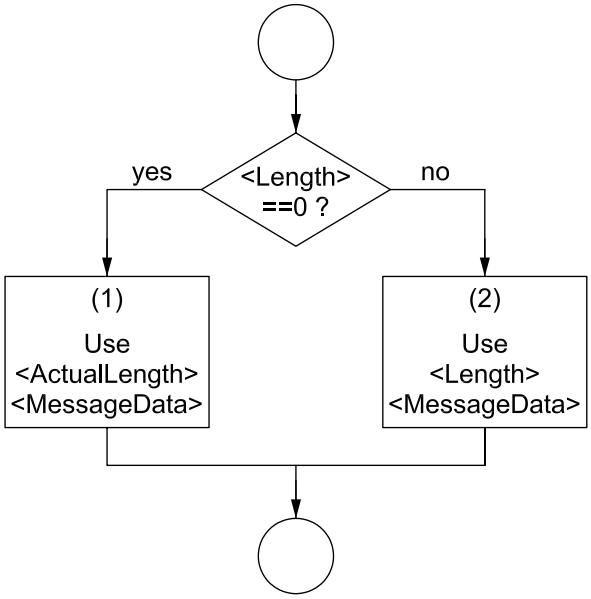


Figure 7 — Handling of <ActualLength> and <Length>

1) Unknown message length transmission (<Length> == 0):

The upper layer provides the amount of data actually available for transmission. The length of this data is given in <ActualLength>. Therefore, the sender shall use the value of the parameter <ActualLength> for the transmission of the Data, but shall indicate in the STF C_PDU an unknown message length transmission by setting ML=0. Further data are fed in during the ongoing transmission, also using <Length>=0 and <ActualLength> indicating the amount of further data. A parameter value <ActualLength>==0 indicates that no further data are available for transmission and the communication layer can transmit the LF C_PDU, which indicates the summed up <ActualLength> values in the message length (ML) also taking into account the data that might be part of the LF C_PDU.

2) Known message length transmission (<Length> != 0):

If the message length is known, the communication layer needs to distinguish the following cases.

a) The communication layer is able to handle the complete amount of data for transmission provided by the upper layer. In this case, the <ActualLength> is set to zero and will not be considered. The message length (ML) included in the STF C_PDU and the LF C_PDU[2)] is equal to <Length>.

b) Due to buffer restraints, the upper layer is not able to handle the complete amount of data to be transmitted although the total message length is known. Therefore, an <ActualLength> which is unequal zero is provided by the upper layer in the initial request service primitive to the communication layer together with the total known message length (ML). The upper layer initiates subsequent calls to the request service primitive to provide further data for the message transmission, where <Actual length> indicates the amount of data provided in the specific request service primitive and the message length (ML) is “don't care”, because it has been provided in the initial request service primitive already.

2) Only applicable in case of a segmented message transfer.

6.3.5 <MessageData>

Type: string of bytes

Range: 0 to <Length> or 0 to <ActualLength>

Description: In the case of a known message length transmission, this parameter includes all the data the upper layer entities exchange. In the case of an unknown message length transmission, the already available data are included.

6.3.6 <StatusParameter>

Type: enumeration

Range: C_COMM_STATE, C_PROGRESS, C_NUM_RETRIES, C_NUM_WAIT

— C_COMM_STATE

This value is used to request the current general status of the ongoing communication.

— C_PROGRESS

This value is used to request progress information on the ongoing communication.

— C_NUM_RETRIES

This value is used to request the number of retries already performed for the ongoing communication.

— C_NUM_WAIT

This value is used to request the number of wait frames already sent for the ongoing communication (reception of multiframe message).

6.3.7 <C_Status_Value>

Description: This parameter contains the status value relating to the requested communication execution. Its type and range depend on the requested status information.

— C_COMM_STATE

Type: enumeration

Range: C_NOT_ACTIVE, C_ACTIVE_RX, C_ACTIVE_TX

— C_PROGRESS

Type: 2 byte unsigned integer value

Range: 0-FFFF hex

For an active receive communication, this value indicates the number of bytes already received. For a transmit communication, this value indicates the number of bytes already transmitted.

— C_NUM_RETRIES

Type: 8 bit unsigned integer value

Range: 0-FF hex

This value indicates the number of retries performed for the ongoing transmission/reception.

— C_NUM_WAIT

Type: 8 bit unsigned integer value

Range: 0-FF hex

This value indicates the number of wait frames received/transmitted for the ongoing transmission/reception.

6.3.8 <Parameter>

Type: enumeration

Range: BC, BfS, MAX_WFT, MAX_RETRIES

This parameter identifies a parameter of the communication layer.

— BC: Parameter used by the receiver in the FC C_PDU.

— BfS: Parameter used by the receiver in the FC C_PDU.

— MAX_WFT: Parameter used by the receiver to specify the maximum number of FC C_PDUs to be transmitted in a sequence with FS set to WAIT (see 7.5.5.2.2).

— MAX_RETRIES: Parameter used by the receiver to specify the maximum number of retries to be performed for a single block of data during an ongoing communication (see 7.5.7.2.3).

6.3.9 <Parameter_Value>

This parameter is assigned to a protocol parameter <Parameter> as indicated in the service section of this document.

— BC

Type: 8 bit unsigned integer value

Range: 0 to FF hex

— BfS

Type: 16 bit unsigned integer value

Range: 0 to FFFF hex

— MAX_WFT

Type: 8 bit unsigned integer value

Range: 0 to FF hex

— MAX_RETRIES

Type: 8 bit unsigned integer value

Range: 0 to FF hex

6.3.10 <Result_ChangeParameter>

Type: enumeration

Range: C_OK, C_RX_ON, C_WRONG_PARAMETER, C_WRONG_VALUE

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

NOTE Table 4 implicitly defines the priority of the result value evaluation by the order of the result values (the first entry has the highest priority).

Table 4 — Result_ChangeParameter values

Result_ ChangeParameter	Sender	Receiver	Description
C_WRONG_PARAMETER	X	X	This value is issued to the service user to indicate that the service did not execute due to an undefined <Parameter>.
C_WRONG_VALUE	X	X	This value is issued to the service user to indicate that the service did not execute due to an out of range or reserved <Parameter_Value>.
C_RX_ON	—	X	This value is issued to the service user to indicate that the service did not execute since a reception of the message identified by <AI> was taking place.
C_OK	X	X	This value means that the service execution has completed successfully.

6.3.11 <Result_GetParameter>

Type: enumeration

Range: C_OK, C_WRONG_PARAMETER

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

NOTE Table 5 implicitly defines the priority of the result value evaluation by the order of the result values (the first entry has the highest priority).

Table 5 — Result_GetParameter values

Result_ GetParameter	Sender	Receiver	Description
C_WRONG_PARAMETER	X	X	This value is issued to the service user to indicate that the service did not execute due to an undefined <Parameter>.
C_OK	X	X	This value means that the service execution has completed successfully.

6.3.12 <Result_GetStatus>

Type: enumeration

Range: C_OK, C_WRONG_PARAMETER

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

NOTE Table 6 implicitly defines the priority of the result value evaluation by the order of the result values (the first entry has the highest priority).

Table 6 — Result_GetStatus values

Result_GetStatus	Sender	Receiver	Description
C_WRONG_PARAMETER	X	X	This value is issued to the service user to indicate that the service did not execute due to an undefined <StatusParameter>.
C_OK	X	X	This value means that the service execution has completed successfully.

7 Communication layer protocol

7.1 Protocol functions

The communication layer protocol performs the following functions:

a) "unacknowledged" and "acknowledged with retry" transmission/reception of messages with a known message length of up to FFFF hex (64 kbyte − 1) bytes;

b) "unacknowledged" and "acknowledged with retry" transmission/reception of messages with an unknown but finite message length of up to FFFF hex (64 kbyte − 1) bytes;

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

7.2 Unsegmented transmission (only start frame)

Transmission of messages up to the available length of an L_PDU is performed via transmission of a single start frame (STF) C_PDU.

Reception of messages up to the available length of an L_PDU is performed via reception of a single start frame (STF) C_PDU.

See Figure 8.

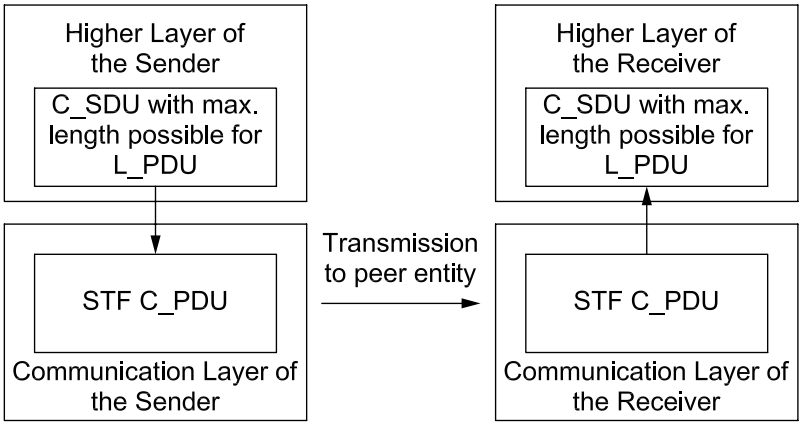


Figure 8 — Example of unsegmented message

NOTE In the case of an acknowledged transmission, the receiver sends back a flow control (FC) C_PDU that acknowledges the reception of the single start frame (STF) C_PDU (not shown in the figure above).

7.3 Segmented transmission (multiple frame transmission)

Transmission of messages which are either longer than the available length of an L_PDU or where the dynamic data transfer makes use of FPL to indicate the amount of included payload data although the total message length might fit into a single L_PDU, are performed via segmentation of the message and transmission of multiple C_PDUs.

Reception of longer messages is performed via reception of multiple C_PDUs and reassembly of the received data bytes (concatenation). The multiple C_PDUs are called StartFrame (for the first C_PDU of the message), ConsecutiveFrame (for the following C_PDUs) as well as LastFrame (concludes a segmented message transmission).

The receiver of a multiple C_PDU message has the possibility of adapting the transmission throughput to its capability by means of the flow control mechanism using the FlowControl protocol data units (FC C_PDU).

Segmented messages are segmented into:

— a StartFrame protocol data unit (STF C_PDU), containing the first byte(s);

— zero or more ConsecutiveFrame protocol data units (CF C_PDU), containing each up to the maximum length of the L_PDU size (might make use of varying FPL values in the case of a dynamic data transfer);

— one LastFrame C_PDU that concludes the transmission and might include the final data of the message.

Figure 9 shows segmentation at the sender side and reassembly at the receiver side.

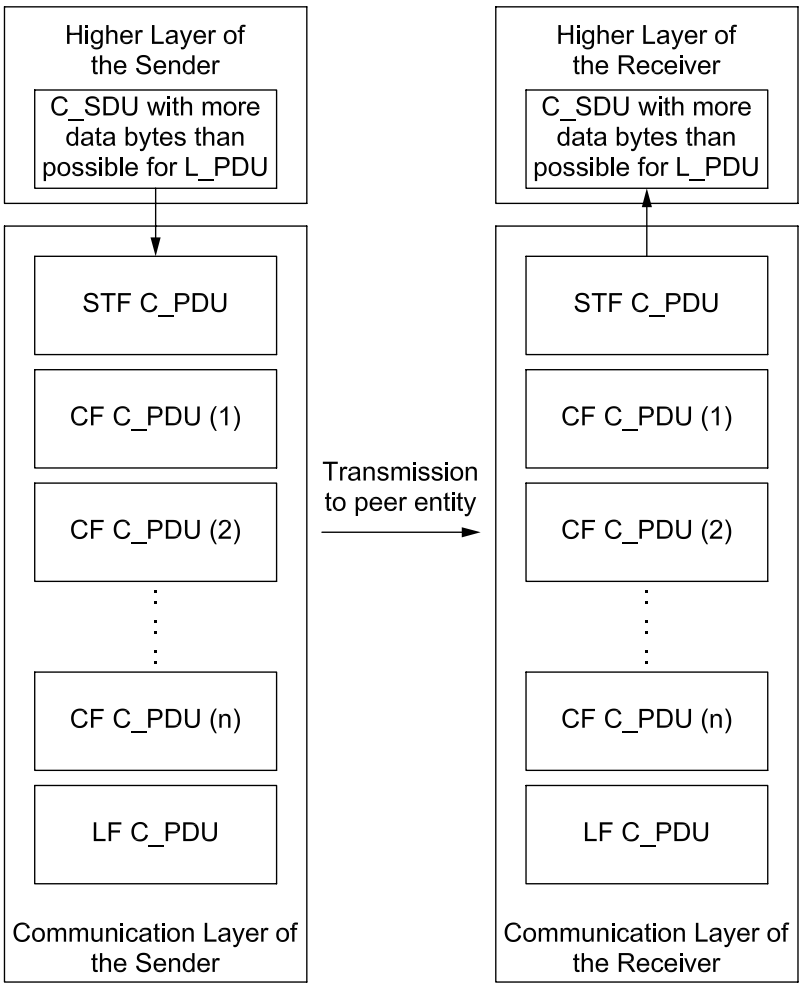


Figure 9 — Example of segmented message

NOTE The example shows the case when the message data do not fit into a single STF C_PDU and the FPL is fixed. The FC C_PDU issued by the receiver in response to the reception of the STF C_PDU is not shown. In the case of an acknowledged transmission, the receiver sends back a flow control (FC) C_PDU that acknowledges the reception of the last CF C_PDU (not shown in Figure 9).

In the case of a known message length transmission, the message length is transmitted in the STF C_PDU. All CF C_PDUs are numbered by the sender to help the receiver reassemble them in the same order.

In the case of an unknown message length transmission, the message length transmitted in the STF C_PDU indicates “0”. A last frame (LF) C_PDU indicates the transmitted amount of data. All CF C_PDUs in between the STF C_PDU and the LF C_PDU are numbered by the sender to help the receiver reassemble them in the same order.

The flow control mechanism (see Figure 10) allows the receiver to inform the sender about the receiver’s capabilities. Since different nodes may have different capabilities, the flow control sent by the receiver informs the sender about its capabilities. The sender shall operate within the receiver’s capabilities.

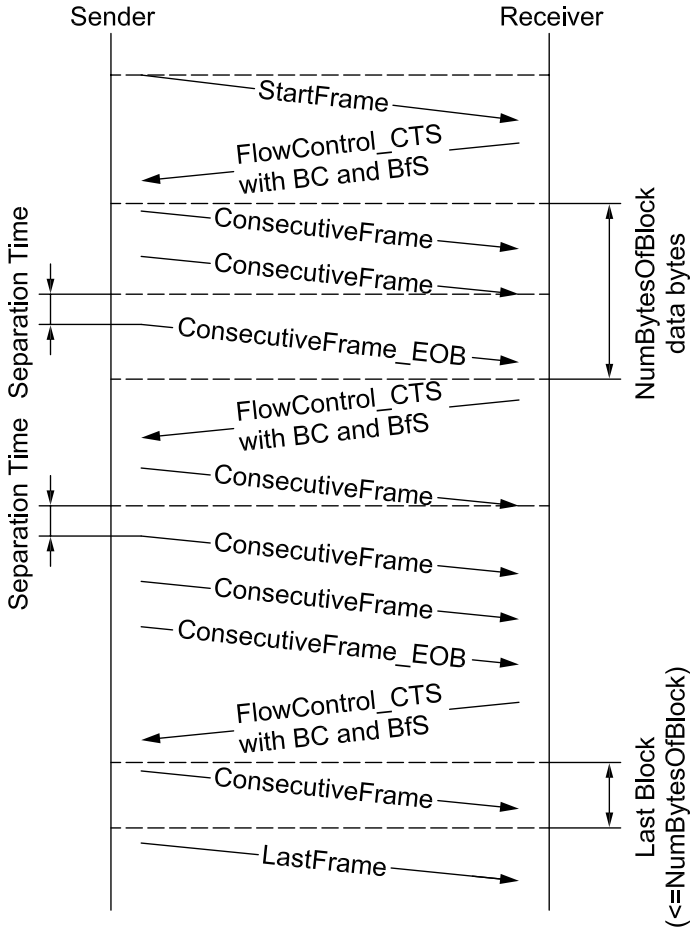


Figure 10 — Flow control mechanism

These capabilities are defined as follows.

— BufferSize (BfS): The maximum number of data bytes the receiver allows the sender to send via CF C_PDUs, before waiting for an authorization to continue transmission of the following data bytes via further CF C_PDUs (see Annex A for implementation example). The sender adapts to the BfS value reported by the receiver and its own buffer size by using the minimum value from both sides as an

indication where the ConsecutiveFrame_EOB has to be transmitted to confirm proper transmission/reception of this data block (for detailed information on BfS, refer to 7.5.5.3)

NumBytesOfBlock = min (reported BfS of receiver, sender buffer size)

— Bandwidth Control (BC): This parameter is composed of subparameters “Separation Cycle Exponent” (SCexp) and “Maximum Number of PDUs Per Cycle” (MNPC).

The associated separation time is calculated based on both subparameters of BC. Therefore, the value of the separation time is either the time between two CF CPUs (or between a final CF C_PDU and LF C_PDU) and/or the Separation Cycles Exponent (SCexp). A detailed definition of Bandwidth Control (BC) is stated in 7.5.5.4

All blocks, except the last one, will consist of the amount of C_PDUs required to transmit the number of data bytes as indicated via NumBytesOfBlock. This is achieved via the transmission of multiple CF C_PDUs where the individual CF C_PDU may or may not be filled to its maximum number of bytes.

Depending on the maximum length of the C_PDU sent prior to a LF C_PDU, the LF may or may not contain data bytes.

Each time the sender/receiver waits for a C_PDU from the receiver/sender, a timeout mechanism provides detection of a transmission failure (see 7.6).

By means of FC C_PDUs, the receiver has the possibility to control transmission of subsequent CF C_PDUs.

— FC.CTS: The authorization to continue

— FC.WAIT: The request to keep waiting

— FC.ABT: Abort the current message transmission/reception

— FC.ACK/RET: Acknowledge the reception of a complete message or request a transmission retry of data out of the current block transmitted

— FC.OVFLW: Buffer overflow, the indication that the message length (ML) specified in the StartFrame of the segmented message exceeds the number of bytes that can be accepted by the receiver entity

There is a maximum limit to the number of FC.WAIT C_PDUs a receiver is allowed to send in a row: C_WFTmax. This parameter is a system design constant and is not transmitted in the FC C_PDU.

7.4 Communication layer protocol data units (C_PDUs)

7.4.1 Protocol data unit types

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

This part of ISO 10681 specifies four different types of communication layer protocol data units:

— start frame (STF C_PDU);

— consecutive frame (CF C_PDU);

— flow control frame (FC C_PDU);

— last frame (LF C_PDU).

These are used to establish a communication path between the peer communication layer entities, to exchange communication parameters, to transmit user data and to release communication resources.

7.4.2 STF C_PDU

The STF C_PDU is identified by the start-frame protocol control information (STF C_PCI). The STF C_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 either unsegmented or segmented messages.

If the STF ML (message length) indicates the same length as the amount of data transferred in this C_PDU, then an unsegmented transmission is performed and this STF concludes the transmission already.

If the STF ML (message length) indicates a length greater than the amount of data transferred in this C_PDU, then a segmented transmission is performed and the receiving communication layer entity shall start assembling the segmented message on receipt of this STF C_PDU.

There are two STF C_PDUs defined in order to distinguish between the following types of transmission.

— STFU_C_PDU: Unacknowledged transmission

— STFA C_PDU: Acknowledged transmission

NOTE If no distinction is required between the types of transmission, this document uses the term “STF” for simplicity. Otherwise, the precise term (“STFU” or “STFA”) is used.

7.4.3 CF C_PDU

The CF C_PDU is identified by the consecutive-frame protocol control information (CF C_PCI). The CF C_PDU transfers segments (C_Data) of the service data unit message data (<MessageData>). All C_PDUs transmitted by the sending entity after the STF C_PDU that indicated a segmented transmission shall be encoded as CF C_PDUs. The message transmission is concluded with a LF C_PDU.

7.4.4 FC C_PDU

The FC C_PDU is identified by the flow-control protocol control information (FC C_PCI). The FC C_PDU instructs a sending network entity to, for example, start, stop or resume transmission of CF C_PDUs.

It shall be sent by the receiving communication layer entity to the sending communication layer entity considering the following conditions for unacknowledged transmission:

a) after the reception of an STFU C_PDU that indicates a segmented message transmission;

b) after the last CF C_PDU of a block of consecutive frames, if further consecutive frames need to be sent.

It shall be sent by the receiving communication layer entity to the sending communication layer entity considering the following conditions for acknowledged transmission:

c) after reception of an STFA C_PDU that indicates segmented or unsegmented message transmission;

d) after the last CF C_PDU of a block of consecutive frames, if further consecutive frames need to be sent;

e) after an LF C_PDU that concludes a message transmission/reception.

The FC C_PDU can also inform a sending network entity to pause transmission of CF C_PDUs during a segmented message transmission or to abort the transmission of a segmented message if, for example, no buffer is currently available to store the message.

7.4.5 LF C_PDU

The LF C_PDU is identified by the last-frame protocol control information (LF C_PCI). The LF C_PDU shall be sent out by the sending network entity to conclude any segmented message transmission and will be received by the receiving network entity. The receiving entity shall pass the assembled message to the service user of the network receiving as follows.

— Unacknowledged transmission: The LF C_PDU has been received.

— Acknowledged transmission: The FC_ACK C_PDU has been transmitted by the receiver following the reception of the LF C_PDU.

This applies to both a known and an unknown message length transmission/reception.

7.4.6 Protocol data unit field description

7.4.6.1 C_PDU format

The protocol data unit (C_PDU) enables the transfer of data between the communication layer in one node and the communication layer in one or more other nodes (peer protocol entities). It has to be distinguished between the following three C_PDU formats. All C_PDUs consist of three fields, as given in Table 7.

Table 7 — C_PDU format

Address information	Protocol control information	Data field
C_AI	C_PCI	C_Data

7.4.6.2 Address information (C_AI)

The C_AI is used to identify the communicating peer entities of the communication layer. The C_AI information received in the C_SDU — C_SA, C_TA, C_TAtype,— shall be copied and included in the C_PDU. If the message data (<MessageData> and <Length>) received in the C_SDU is so long that segmentation is needed for the communication layer to transmit the complete message, the C_AI shall be copied and included (repeated) in every C_PDU that is transmitted. This field 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 6.3.1.

7.4.6.3 Protocol control information (C_PCI)

This field identifies the type of C_PDUs (byte 1) and is also used to exchange other control parameters (from byte 2 on) between the communicating communication layer entities.

NOTE For a detailed specification of all C_PCI parameters, see Table 8.

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

The size of C_Data depends on the C_PDU type and the assigned frame payload length (FPL).

7.5 Protocol control information specification

7.5.1 General

This subclause defines the Protocol Control Information of the communication layer. The error handling for the applicable parameters is defined in 7.5.7.

7.5.2 C_PCI

Each C_PDU is identified by means of a C_PCI as defined in Table 8 and Table 9.

Table 8 — Summary of C_PCI bytes

Name	Byte 1		Byte 2	Byte 3	Byte 4
	Bits 7-4 C_PCIType	Bits 3-0			
StartFrame (STFU)	4	0	FPL	ML	
StartFrame_ACK (STFA)	4	1	FPL	ML	
ConsecutiveFrame_1 (CF1)	5	SN	FPL		
ConsecutiveFrame_2 (CF2)	6	SN	FPL		
ConsecutiveFrame_EOB (CFEOB)	7	SN	FPL		
FlowControl (FC)	8	FS=CTS	BC	BfS	
FlowControl (FC)	8	FS=ACK_RET	ACK	BP	
FlowControl (FC)	8	FS=WT			
FlowControl (FC)	8	FS=ABT			
FlowControl (FC)	8	FS=OVFLW			
LastFrame (LF)	9	0	FPL	ML	

NOTE Grayed out cells are not utilized for PCI information, but depending on the PDU, they might be utilized for payload data

Table 9 — Definition of C_PCItype values

Hex value	Description
0 to 3	**Reserved** This range of values is reserved by ISO 10681.
4	**StartFrame / StartFrame_ACK** A StartFrame (STFU/STFA) shall be used to support the transmission of unsegmented or segmented messages. In the case of an unsegmented transmission, the corresponding StartFrame includes the complete message and can be identified by the Message Length (ML) being equal to the length of the data content of the StartFrame. In the case of a segmented transmission, the corresponding StartFrame includes the first part of the message and can be identified by the Message Length (ML) being greater than the length of the data content of the StartFrame. On receipt of a StartFrame, the receiving communication layer entity shall start assembling the segmented message. For an unacknowledged transmission, the low nibble includes the value “0” and for an acknowledged transmission the low nibble includes the value “1”.
5	**ConsecutiveFrame_1** When sending segmented data, all consecutive frames following the STFU/STFA are encoded as ConsecutiveFrame (CF). On receipt of a CF, the receiving communication layer entity shall assemble the received data bytes until the whole message is received. The receiving entity shall pass the assembled message to the upper layer either after the last frame of the message has been received without error (unacknowledged transmission) or after the transmission of the acknowledgement of the message reception (acknowledged transmission). For the purpose of a retry mechanism, the C_PCItype definition differentiates between two types of ConsecutiveFrames (for detailed definitions, see 7.5.7.2.3). In a case where no retry is supported, ConsecutiveFrame_1 shall always be used for the whole message transmission. A segmented message transmission shall always start with ConsecutiveFrame_1.
6	**ConsecutiveFrame_2** The format of ConsecutiveFrame_2 is identical to the definitions of ConsecutiveFrame_1. The usage is according the definitions in 7.5.7.2.3.

Table 9 (*continued*)

Hex value	Description
7	**ConsecutiveFrame_EOB** If there is a need to actively acknowledge a block of transmitted data (due to buffer restraints in either the sender or the receiver), the sender shall use the ConsecutiveFrame_EOB (EOB=End Of Block) to actively request the receiver to acknowledge the reception of a block of transmitted data. This mechanism is, for example, used when the sender has restricted buffer resources and a message can only be transmitted in certain blocks and a retry is used in order to make sure that the buffer can be released and can be used for the next block of data for the message to be transmitted.
8	**FlowControl** The purpose of FlowControl (FC) is, for example, to negotiate the buffer capabilities of the receiver and to request retry of transmission within a current block of data. Five distinct types of 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 defined in 7.5.5.2.
9	**LastFrame** The last frame is used by the sender to conclude any type of transmission (known and unknown message length). The message length (ML) parameter indicates the total length of the message, not taking into account bytes redundantly transmitted during retries. For a known message length transmission, the indicated message length is therefore identical to the message length given in the StartFrame.
A to F	**Reserved** This range of values is reserved by ISO 10681.

7.5.3 StartFrame / StartFrame_ACK C_PCI parameter definition

Table 10 provides an overview of the STFU/STFA C_PCI byte.

Table 10 — Overview of STFU/STFA C_PCI bytes

Name	Byte 1		Byte 2	Byte 3	Byte 4
	Bits 7-4 C_PCIType	Bits 3-0			
StartFrame (STFU)	4	0	FPL	ML	
StartFrame_ACK (STFA)	4	1	FPL	ML	

The parameter FPL (Frame Payload Length) is used in the STF C_PDU to specify the number of payload data bytes included in this STF C_PDU. The length of the ML (Message Length) parameter is not considered in the FPL value (see Table 11).

Table 11 — Definition of FPL values

Hex value	Description
0	**Not allowed**
	These values are not allowed as FPL values in the STF.
1 to F6[a]	**Frame Payload Length**
	This value reflects the amount of payload data bytes included in this STF C_PDU.
F7 to FF	**Not allowed**
	These values are not allowed as FPL values in the STF.

[a] The minimum value is based on the fact that at least a single payload data byte is included in the STF/STFA. The maximum value F6 hex is based on the maximum amount of data that can be placed in a single L_PDU (127 words = 254 bytes), minus the address information (4 bytes) and minus the C_PCI (4 bytes).

The parameter ML (Message Length) is used in the STF C_PDU to specify the number of service user data bytes (see Table 12).

Table 12 — Definition of ML values

Hex value	Description
0	**Unknown message length**
	This value indicates that an unknown, but finite message length (max 64 kB − 1) transmission will take place. The amount of data transmitted is finally indicated in the last frame (LF) C_PDU at the end of the transmission.
1 to FFFF	**Message Length**
	This value reflects the amount of service user data bytes. This value is reflected in the last frame (LF) C_PDU at the end of the transmission.

7.5.4 ConsecutiveFrame / ConsecutiveFrame_EOB C_PCI parameter definition

7.5.4.1 CF / CF_EOB C_PCI bytes

Table 13 provides an overview of the CF C_PCI bytes.

Table 13 — Overview of CF C_PCI bytes

Name	Byte 1		Byte 2
	Bits 7-4 C_PCIType	Bits 3-0	
ConsecutiveFrame_1 (CF1)	5	SN	FPL
ConsecutiveFrame_2 (CF2)	6	SN	FPL
ConsecutiveFrame_EOB (CF_EOB)	7	SN	FPL

Table 14: The parameter FPL (Frame Payload Length) is used in the CF C_PDU to specify the number of payload data bytes included in this CF C_PDU.

Table 14 — Definition of FPL values

Hex value	Description
0	**Not allowed**
	This value is not allowed as an FPL value.
1 to F8[a]	**Frame Payload Length**
	This value reflects the amount of payload data bytes included in this CF C_PDU.
F9 to FF	**Not allowed**
	These values are not allowed as FPL values.

[a] The maximum value, F8 hex, is based on the maximum amount of data that can be placed in a single L_PDU (127 words = 254 bytes), minus the address information (4 bytes) and minus the C_PCI information (2 bytes).

7.5.4.2 SequenceNumber (SN) parameter definition

The parameter SN is used in the CF/CF_EOB C_PDU to specify the order of the consecutive frames.

— The SN shall start with zero for all segmented messages. The STFU/STFA shall be assigned the value of zero. It does not include an explicit SequenceNumber in the C_PCI field but shall be treated as the segments number zero.

— The SN of the first CF that immediately follows the STF shall be set to one.

— The SN shall be incremented by one for each new CF that is transmitted during a segmented message transmission.

— The SN value shall not be affected by any FC frame except when a retry is indicated (FS = ACK_RET). Then SN is set to zero.

— When the SN reaches the value of fifteen, it shall wraparound and be set to zero for the next CF.

This shall lead to the sequence given in Table 15.

See Table 16 for SN values.

Table 15 — Example for a sequence of SN values in a segmented transmission

C_PDU	STF	CF	CF	CF	CF	CF	CF	CF
SN (hex)	(0)	1	…	E	F	0	1	…

Table 16 — Definition of SN values

Hex value	Description
0 to F	**SequenceNumber (SN)**
	The SequenceNumber (SN) shall be encoded in the lower nibble bits of C_PCI byte #1. The SN shall be set to a value within the range of zero to fifteen. The SequenceNumber will be reset to zero in the case of a retry.

7.5.5 FlowControl C_PCI parameter definition

7.5.5.1 FC C_PCI bytes

Table 17 provides an overview of the FC C_PCI bytes.

Table 17 — Overview of FC C_PCI bytes

Name	Byte 1		Byte 2	Byte 3	Byte 4
	Bits 7-4 C_PCIType	Bits 3-0			
FlowControl (FC)	8	FS=CTS	BC	BfS	
FlowControl (FC)	8	FS=ACK_RET	ACK	BP	
FlowControl (FC)	8	FS=OVFLW			
FlowControl (FC)	8	FS=WT			
FlowControl (FC)	8	FS=ABT			
NOTE Grayed out cells are not utilized for PCI information.					

7.5.5.2 FlowStatus (FS) parameter definition

7.5.5.2.1 General

The parameter FlowStatus (FS) indicates, for example, whether or not the sending network entity can proceed with the message transmission. See Table 18.

A sending network entity shall support all specified (not reserved) values of the FS parameter.

Table 18 — Definition of FS values

Hex value	Description
0 to 2	**Reserved**
	This range of values is reserved by ISO 10681.
3	**ContinueToSend (CTS)**
	The FlowControl ContinueToSend parameter shall be encoded by setting the lower nibble of the C_PCI byte #1 to “3”. It shall cause the sender to resume the sending of consecutive frames. This value indicates that the receiver is ready to receive a maximum BfS number of data bytes via one or multiple consecutive frames, where each ConsecutiveFrame has to be separated by the sender using BC.
4	**Acknowledge/Retry (ACK_RET)**
	The FlowControl Acknowledge/Retry parameter shall be encoded by setting the lower nibble of the C_PCI byte #1 to "4". The ACK parameter shall be set to the corresponding value, either indicating an acknowledgement or a request for a retry. In the case of a request for a retry, the BP parameter includes the byte position where the sender shall start the retry. This has to be the first wrong byte position. Furthermore, this position shall be within the current block of the transmission.
5	**Wait (WT)**
	The FlowControl Wait parameter shall be encoded by setting the lower nibble of the C_PCI byte #1 to “5”. It shall cause the sender to continue to wait for a new FlowControl C_PDU and to restart its C_BS timer (7.6).
6	**Abort (ABT)**
	The FlowControl Abort parameter shall be encoded by setting the lower nibble of the C_PCI byte #1 to “6”. It shall cause the sender to abort the transmission of the currently ongoing transmission. A possible reason for an abort of a transmission is that the receiver is currently busy and cannot take the request at that point in time; therefore, the transmission of the segmented message was aborted.
7	**Overflow (OVFLW)**
	The FlowControl Overflow parameter shall be encoded by setting the lower nibble of the C_PCI byte #1 to “7”. This value is used to indicate that the buffer on the receiver side of a segmented message transmission cannot store the number of bytes specified by the StartFrame MessageLength (STF_ML) parameter in the StartFrame, thereby aborting the transmission of the segmented message. It causes the sender to abort the currently ongoing transmission.
8 to F	**Reserved**
	This range of values is reserved by this part of ISO 10681.

Table 19 describes the usage of the FlowStatus values of either an acknowledged segmented or unsegmented transmission.

Table 19 — FS value usage

	Communication scenario	
Flow Status	**Segmented transmission (acknowledge/not acknowledged)**	**Unsegmented transmission[a] (acknowledged)**
CTS	As defined for CTS during a multiframe transmission/reception (see Table 18)	N/A
ACK_RET	Used for retry handling of a segmented message transmission (see Table 18)	Used to indicate that the unsegmented transmission can currently not be accepted and the STFA transmission has to be repeated by the sender communication layer. The retry of the STFA is handled within the communication layer and not indicated to the upper layer. If the number of retries has elapsed or a FC C_PDU timeout occurs, the transmission is terminated and the error is indicated to the upper layer. The FC parameter BP has to point to the first data byte of the received STFA (value “1”) and the parameter ACK has to be set to RET (Retry Request).
WT	As defined for WT during a segmented message transmission/reception (see Table 18)	Used to indicate that the STFA has been received correctly, but the data are not yet processed in the CL in order to provide the proper final acknowledgement (ACK) or overflow (OVFLW). The same handling as for a segmented data transmission applies, except that the final FC C_PDU following the last FC C_PDU with FlowStatus WT shall only use FS values as applicable for unsegmented acknowledged transmission.
ABT	As defined for ABT during a segmented message transmission/reception (see Table 18)	N/A
OVFLW	As defined for OVFLW during a segmented message transmission reception (see Table 18)	Used to indicate that the STFA has been received correctly but cannot be processed at all at the current point in time due to memory constraints. Therefore, it has been rejected. The communication layer in the sender terminates the transmission and indicates the error to the upper layer.

[a] A FlowControl C_PDU transmission is not applicable for an unsegmented unacknowledged communication.

7.5.5.2.2 Maximum number of FC.Wait frame transmissions (C_WFTmax)

The purpose of this counter is to avoid communication sender nodes being potentially hooked-up in the 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 protocol data unit.

— The C_WFTmax counter shall indicate how many FC C_PDU WTs can be transmitted by the receiver in a row.

— The C_WFTmax counter upper limit shall be user defined at system generation time.

— The C_WFTmax counter shall only be used on the receiving network entity during message reception.

— If the C_WFTmax counter value is set to zero, then flow control shall rely upon flow control continuing to send FC C_PDU CTS only. Flow control wait (FC C_PDU WT) shall not be used by that network entity.

— The C_WFTmax counter shall be set to its initial value any time a FlowControl frame with FS=CTS is transmitted by the receiver of the segmented message, which means that for any block of ConsecutiveFrames the full range of allowed wait frames is possible.

7.5.5.3 BufferSize (BfS) parameter definition

The BfS parameter shall be encoded in bytes #3 to #4 of the FC C_PCI.

The resolution of BfS is 1 byte per count, representing the number of bytes that can be transferred via CF C_PDUs per block (see Table 20).

The BfS parameter can vary during the transmission of a segmented message, but shall be kept constant by the sending network entity for the duration of a block of ConsecutiveFrames (including the LastFrame). Furthermore, the value of BfS reported by the receiver can be greater than the total message length (ML) stated in the start frame (STF).

Figure 11 shows an example for of buffer size handling “BfS” (see also 7.3) where the receiver provides a total buffer size of 148 bytes.

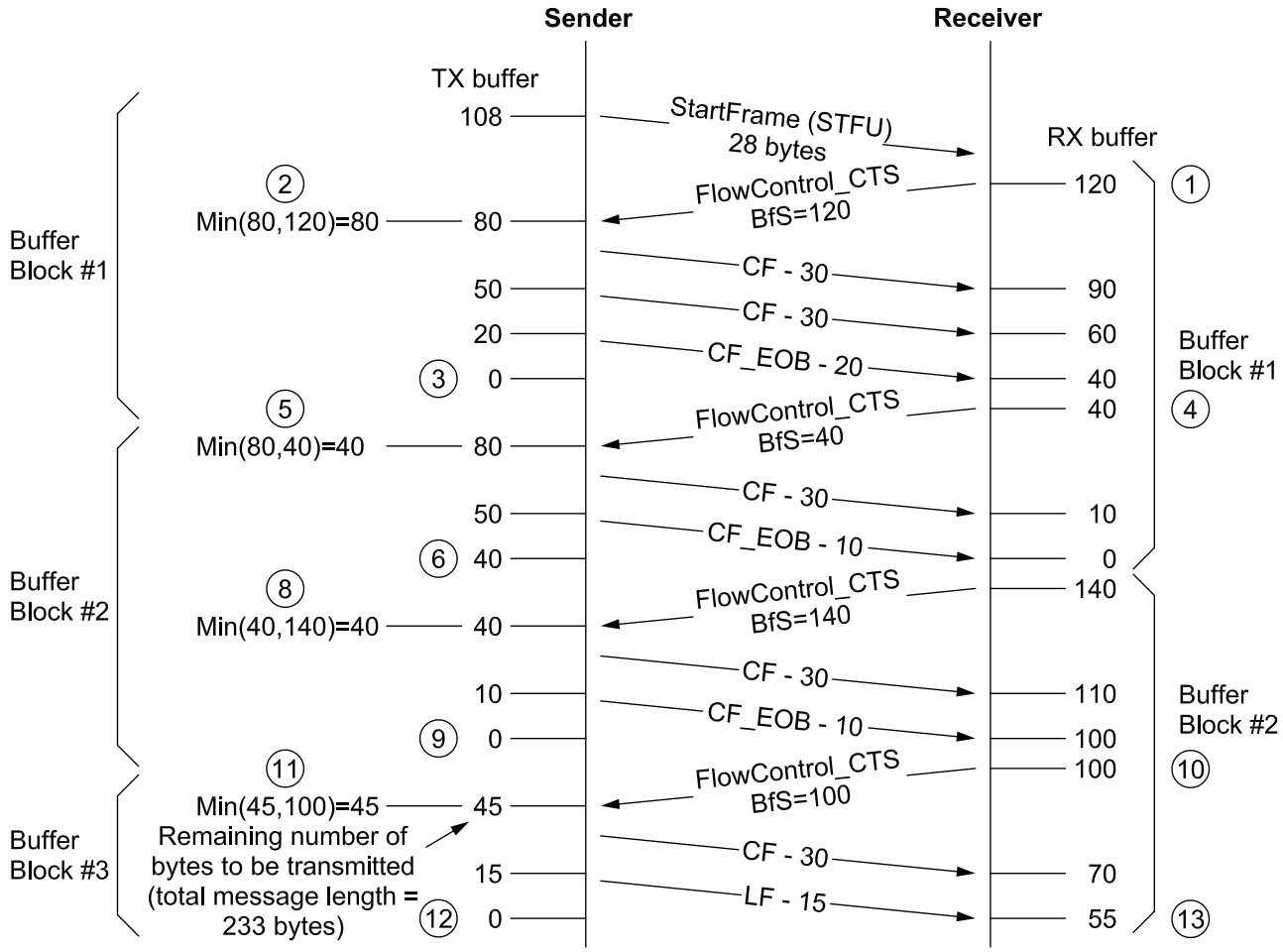


Figure 11 — Example for BfS handling

1) The receiver indicates to the sender that it has a buffer for 120 bytes for the next block of data.

2) The sender determines the amount of data it can transmit based on the reported buffer capability of the receiver and its own buffer of "108 – STFU_length = 80 bytes" (result = 80 bytes). Following that, the sender transmits the first ConsecutiveFrame of the block of data.

3) The last ConsecutiveFrame (EOB) of this block includes 20 bytes, which is the remaining number of bytes of the sender transmit buffer.

4) The receiver has 40 bytes left in its receive buffer after the reception of the ConsecutiveFrame that concludes the block (EOB); therefore, it reports this amount of buffer in the FlowControl frame BfS value.

5) The sender requests internally a buffer for transmission and retrieves a buffer of 80 bytes in length. The sender determines the amount of data it can transmit based on the reported buffer capability of the receiver and its own buffer of 80 bytes (result = 40 bytes). Following that, the sender transmits the first ConsecutiveFrame of the block of data.

6) The last Consecutive Frame (EOB) of this block includes 10 bytes, which is the remaining number of bytes the receiver can receive based on the reported BfS value for this block of data. The sender has still 40 bytes left for transmission, which has to be postponed to the next block of data to be transmitted.

7) The receiver now requests a new buffer for reception and reports this amount of buffer in the FlowControl frame to the sender.

8) The sender determines the amount of data it can transmit based on the reported buffer capability of the receiver and its own remaining buffer of 40 bytes (result = 40 bytes). Following that, the sender transmits the first ConsecutiveFrame of the block of data.

9) The last ConsecutiveFrame (EOB) of this block includes 10 bytes, which is the remaining number of bytes of the sender transmit buffer.

10) The receiver has 100 bytes left in its receive buffer after the reception of the ConsecutiveFrame that concludes the block (EOB); therefore, it reports this amount of buffer in the FlowControl frame BfS value.

11) The sender requests internally a buffer for transmission of the remaining message data and retrieves a buffer of 45 bytes in length. The sender determines the amount of data it can transmit based on the reported buffer capability of the receiver and its own buffer of 45 bytes, which is the remaining number of bytes of the message to be transmitted (result = 45 bytes). Following that, the sender transmits the first ConsecutiveFrame of the block of data.

12) The last ConsecutiveFrame (EOB) of this block includes 15 bytes, which is the remaining number of bytes of the sender transmit buffer.

13) The receiver has 55 bytes left in its receive buffer after the reception of the ConsecutiveFrame that concludes the block (EOB) and the message, but now the message is completely received and indicated to the upper layer, so the remaining bytes are not used and freed afterwards.

Table 20 provides an overview of the BfS parameter.

Table 20 — Definition of BfS values

Hex value	Description
0	**BufferSize (BfS)**
	The BfS parameter value “0” shall be used to indicate to the sender that no more FC frames shall be sent during the transmission of the segmented message. The sending communication layer entity shall send all remaining consecutive frames without any stop for further FC frames from the receiving communication layer entity.
1 to FFFF	**BufferSize (BfS)**
	This range of BfS parameter values shall be used to indicate to the sender the maximum number of data bytes that can be received without an intermediate FC frame from the receiving network entity.

7.5.5.4 Definition of Bandwidth Control (BC) parameter

By means of the BC parameter, the receiver reports its receiving performance level to the sender, i.e. the maximum bandwidth it can receive. The BC parameter contains the following parameters:

— separation cycle exponent (SCexp);

— maximum number of PDUs per cycle (MNPC).

See Table 21.

Table 21 — Definition of BC values

Hex value		Description
Bit 7-3 MNPC	**Bit 2-0 SCexp**	
0	don't care	**No bandwidth control** If no bandwidth control mechanism is required, bit 7-3 is set to zero.
1 to 1F	0 to 7	**SCexp** The sub-parameter “Separation Cycle Exponent” represents the exponent to calculate the minimum number of “Separation Cycles” (SC) the sender has to wait for the next transmission of a C_PDU. Formula: SC = $(2^n) - 1$ (with n = SCexp) which results in the following separation cycles: 0, 1, 3, 7, 15, 31, 63, 127 **MNPC** The sub-parameter “Maximum Number of PDUs per Cycle” limits the number of C_PDUs the sender is allowed to transmit within a FlexRay cycle either immediately following FC C_PDU or following SC cycles after the sender has sent the previous C_PDU of the message. The adherence of NMPC prevents an overflow on the receiver side (e.g. due to Rx-buffer restraints in the receiver). Note: Both parameters can be set independently.

Figure 12 graphically depicts how SCexp and MNPC are evaluated during a segmented message transmission. The example given in the figure assumes that the receiver is capable of receiving three CF C_PDUs within a cycle (MNPC = 3) but requires a separation of cycles of three (SCexp = 2).

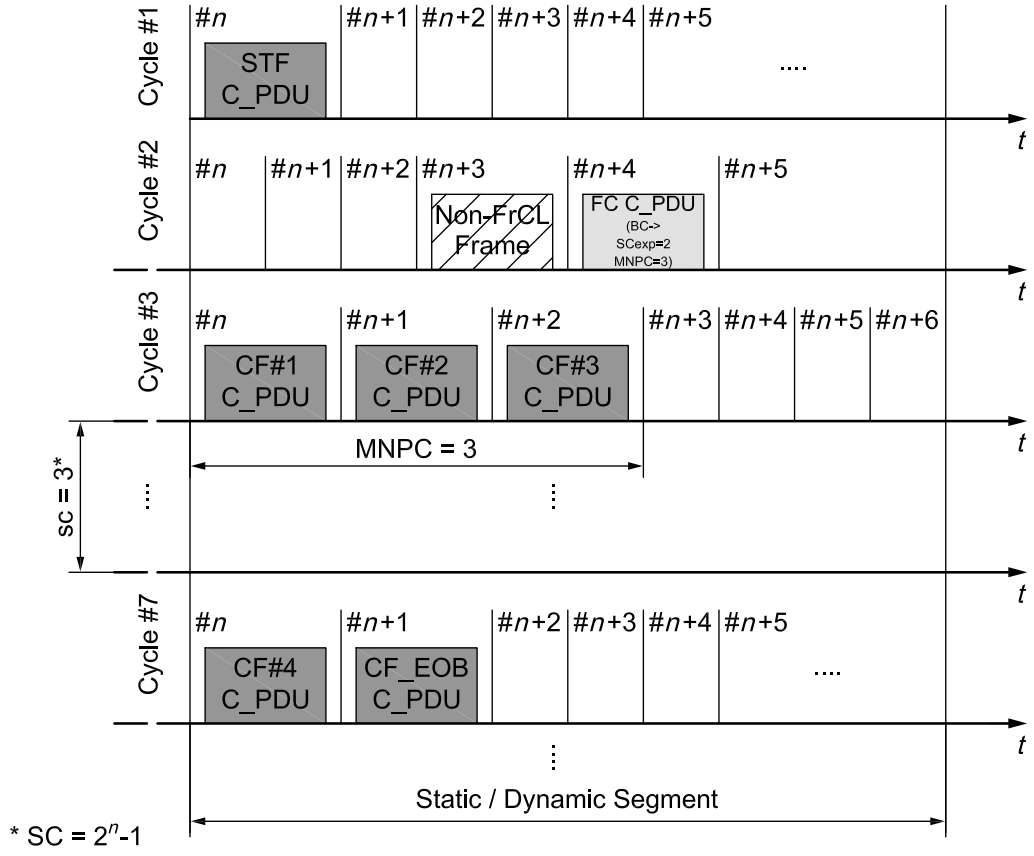


Figure 12 — Example of SCexp and MNPC handling

NOTE If during a given FlexRay-cycle #x, the CL sends at least one C_PDU but less than the number of C_PDUs allowed by MNPC, then the next C_PDU of that connection may not be sent before cycle #(x + SC + 1).

7.5.5.5 Acknowledge (ACK) parameter definition

The parameter ACK (Acknowledge) is used in the FC C_PDU to conclude a successful transmission or to request a retry in the case of an SN sequence error (see Table 22).

Table 22 — Definition of ACK values

Hex value	Description
0	**Acknowledge (ACK)**
	In the case of an acknowledged transmission, this value indicates that the message has been received completely by the receiver without any error. The value of the parameter BP (Byte Position) is “don't care” in this situation, but it is recommended to set it to zero.
1	**Retry Request (RET)**
	In the case of an acknowledged transmission, this value indicates that the receiver has detected an SN sequence error and requests a retry from the first wrong byte position, which is indicated via the Byte Position (BP) parameter (see 7.5.7.2.3).
2 to FF	**Reserved**
	This range of values is reserved by this part of ISO 10681.

7.5.5.6 Byte Position (BP) parameter definition

The parameter BP (Byte Position) is used in the FC C_PDU to specify the first wrong byte position where the sender shall start the retry in the case of an SN sequence error (see Table 23).

Table 23 — Definition of BP values

Hex value	Description
0 to FFFE	**Byte Position (BP)**
	This value indicates the position within a block of consecutive frames where the sender shall start the retransmission. This value is usually the first wrong byte position detected by the receiver. The value starts at "0" with each block of consecutive frames. The value of BP must be within the current block of data. The utilization of Byte Position refers to the acknowledge and retry mechanism caused by an incorrect sequence number (SN) detection (see 7.5.7.2).
FFFF	**Reserved**
	This value is reserved by this part of ISO 10681.

7.5.6 LastFrame C_PCI parameter definition

Table 24 provides an overview of the LF C_PCI byte.

Table 24 — Overview of LF C_PCI byte

Name	Byte 1		Byte 2	Byte 3	Byte 4
	Bits 7-4 C_PCIType	Bits 3-0			
LastFrame (LF)	9	0	FPL	ML	

The parameter ML (Message Length) is used in the LF C_PDU to specify the number of service user data bytes at the end of an unknown message length transmission.

The parameter FPL (Frame Payload Length) is used in the LF C_PDU to specify the number of payload data bytes included in this LF C_PDU. See Table 25.

Table 25 — Definition of FPL values

Hex value	Description
0 to F6[a]	**Frame Payload Length**
	This value reflects the amount of payload data bytes included in this LF C_PDU.
F7 to FF	**Not allowed**
	These values are not allowed as FPL values.

[a] The maximum value "F6 hex" is based on the maximum amount of data that can be placed in a single L_PDU (127 words = 254 bytes), minus the address information (4 bytes) and minus the C_PCI information (4 bytes).

Table 26 — Definition of ML values

Hex value	Description
0	**Reserved**
	This value is reserved by ISO 10681.
1 to FFFF	**Message Length**
	This value reflects the amount of service user data bytes which have been transmitted in this message, not taking into account any performed retry. In the case of a known message length transmission, this message length is identical to the ML parameter in the STF C_PDU.

7.5.7 Error handling

7.5.7.1 ML and FPL error handling

In the case of known message length transmission, the following error handling applies. See Table 27.

Table 27 — ML/FPL error handling for known message length

	Segmented		Unsegmented	
	Sender	**Receiver**	**Sender**	**Receiver**
Unacknowledged	N/A	($ML_{STF} \neq ML_{LF}$) OR ($ML_{STF} \neq \sum FPL$) OR ($ML_{LF} \neq \sum FPL$): Discard received data and generate C_Data.indication with C_Result = C_ML_MISMATCH.	N/A	FPL > ML OR FPL > FPL_{max} Ignore
Acknowledged	Transmission is aborted once FC_ABORT is received and a C_Data.confirm with C_Result = C_ABORT.	($ML_{STF} \neq ML_{LF}$) OR ($ML_{STF} \neq \sum FPL$) OR ($ML_{LF} \neq \sum FPL$): Discard received data and generate C_Data.indication with C_Result = C_ML_MISMATCH and transmit a FC C_PDU with FC_ABORT.	Transmission is aborted once FC_ABORT is received and a C_Data.confirm with C_Result = C_ABORT.	FPL > ML OR FPL > FPL_{max}: Ignore

In the case of an unknown message length transmission, the following error handling applies. See Table 28.

Table 28 — ML/FPL error handling for unknown message length

	Segmented		Unsegmented	
	Sender	**Receiver**	**Sender**	**Receiver**
Unacknowledged	N/A	$ML_{LF} \neq \sum FPL$: Discard received data and generate C_Data.indication with C_Result = C_ML_MISMATCH.	N/A	N/A
Acknowledged	Transmission is aborted once FC_ABORT is received and a C_Data.confirm with C_Result = C_ABORT.	$ML_{LF} \neq \sum FPL$: Discard received data and generate C_Data.indication with C_Result = C_ML_MISMATCH and transmit an FC C_PDU with FC_ABORT.	N/A	N/A

7.5.7.2 SN error handling

7.5.7.2.1 General

If a CF C_PDU message is received with an incorrect sequence number, then error handling as defined in this subclause has to be applied. It has to be distinguished between unacknowledged and acknowledged transmission.

7.5.7.2.2 Unacknowledged message transmission

The message reception shall be aborted, and the communication layer shall make a C_Data.indication service call with the parameter <C_Result> = C_WRONG_SN to the upper layer.

7.5.7.2.3 Acknowledged message transmission and retry mechanism

7.5.7.2.3.1 General

Retry mechanism is a part of error handling within the communication layer. The retry mechanism is only possible within acknowledged message transmission. If a retry is necessary, the receiver signals a retry request by sending FlowControl_ACK_RET (FC_ACK_RET). The parameter BufferPosition (BP) of FC_ACK_RET signals the exact position the sender shall restart data transmission. This position is not necessarily equal with the start of a block. Up to the last acknowledged byte position, each value is possible.

A retry may be requested in any situation during the communication. It may be initiated after an STFA had been received, during a segmented message transmission.

— A retry after an unsegmented/segmented message transfer (so far only STFA has been sent) may occur due to the situation that the upper layer has temporarily no buffer available to even store the length of an STF.

— If a retry is initiated while a segmented message transmission is ongoing, the first retry requests the sender to switch from ConsecutiveFrame_1 type to ConsecutiveFrame_2 type. If more than one retry is requested within a segmented transmission, the Consecutive_Frame type toggles.

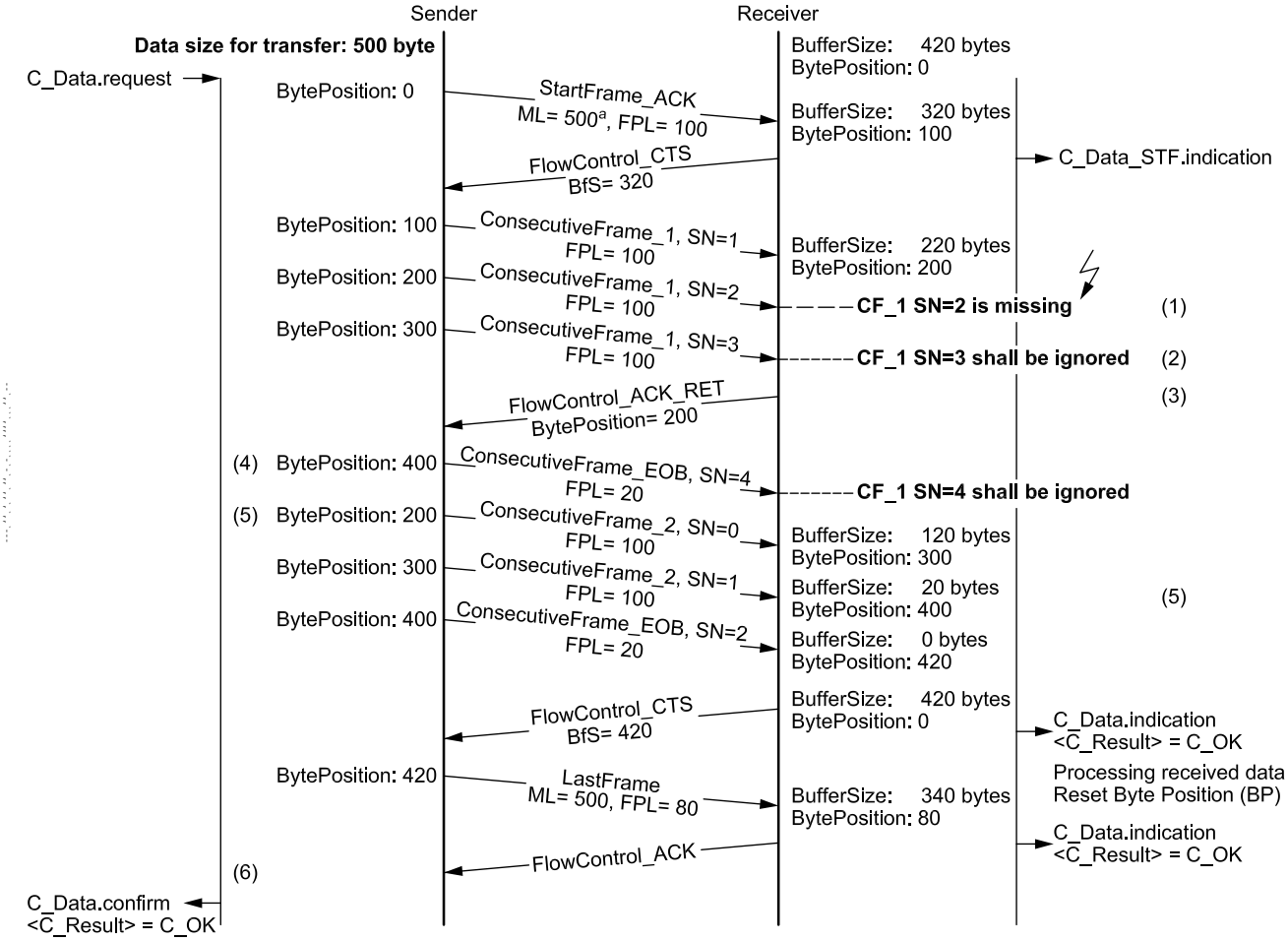
The maximum number of retries within a segmented transmission is restricted to a configurable value which resides in the communication layer on the receiver side.

If the maximum number of retries within a segmented transmission has been reached, the receiver shall abort the transfer by sending FlowControl_ABORT.

The sender shall not reset all internal information (e.g. BytePosition start value of the current block) within an acknowledged transfer until a FlowControl_ACK or a FlowControl ABORT has been received.

7.5.7.2.3.2 Retry in the case of an SN error

If there is a sequence number (SN) error, the retry mechanism can be used for an acknowledged transmission with known and unknown message length. Figure 13 shows the retry handling in the case of a sequence error in the received sequence number. For the example given, the CF with SN = 2 is missing in the receiver, which causes a sequence error when the CF with SN = 3 is received.



[a] In the case of an unknown message length transmission, ML is zero.

Figure 13 — Retry in the case of an SN error

The handling is as follows.

1) The receiver misses the CF with SN = 2.

2) The receiver determines an SN sequence error, because the sequence of received SN values is “1” followed by “3”.

3) The receiver indicates the failure and requests a retry via the FC C_PDU with FS=ACK_RET and the byte position (BP) where the retry shall start (the first wrong byte position). The FlowControl is transmitted immediately right after the detection of the error. Sending FlowControl shall not be delayed until CF_EOB is received.

4) For this example, the sender is still in the mode of transmitting CF C_PDUs of the actual block and has not yet evaluated the FC that indicates the retry request. Therefore, the evaluation of the FC might happen before CF with SN = 4 is transmitted. However, the receiver ignores this C_PDU.

5) The sender has evaluated the FC that indicates the retry request and restarts the transmission at the indicated byte position. The sender toggles the CF C_PCI (in the example given to CF_2) and also resets the SN to zero. If further retry occurs, the CF C_PCI is toggled again (to CF_1) and the SN is reset to zero. In general, a toggling of the CF C_PCI occurs any time a retry occurs.

6) The transmission shall be finalized by sending FlowControl_ACK (Acknowledged transmission).

7.5.7.2.3.3 Retry in the case of an ML error

The retry mechanism in the case of a message length (ML) error is generally not allowed. Message length error handling is described in 7.5.7.1.

For all cases, the receiver shall abort the transfer by sending FlowControl_ABORT. Figure 14 shows that scenario.

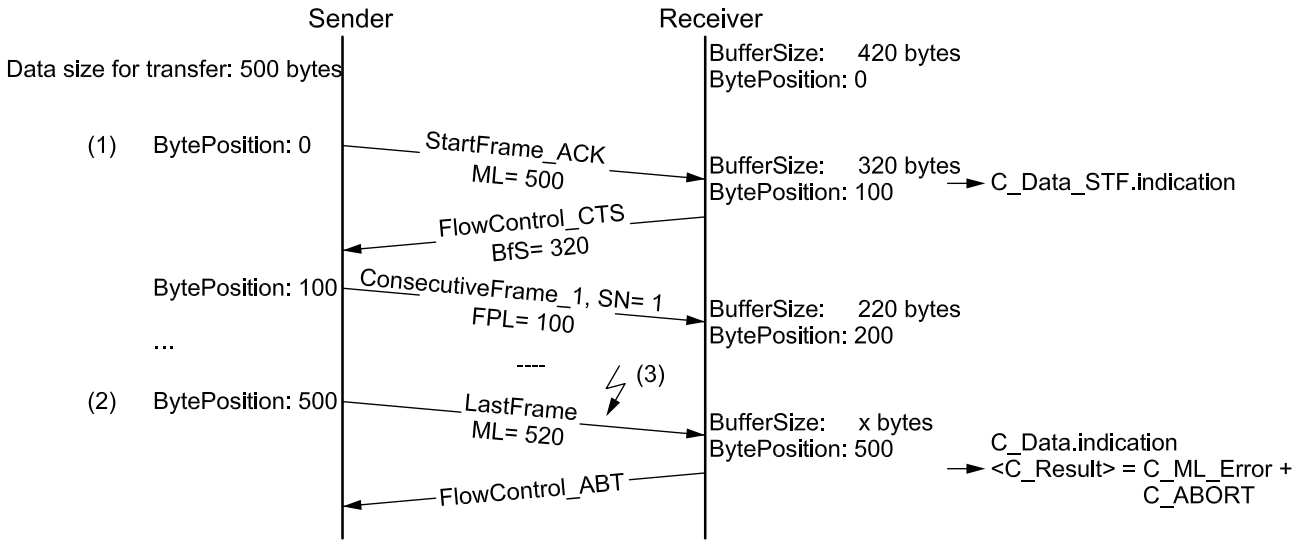


Figure 14 — Handling of an ML error

The handling is as follows:

1) the sender indicates a message length of 500 bytes within StartFrame;

2) the sender indicates that a message length of 520 bytes has been transmitted;

3) the receiver identifies a mismatch of message length and sends FlowControl_ABT.

7.5.7.2.3.4 Result values in the case of a retry

Table 29 shows the different values of <C_Result> in the case of a retry.

Table 29 — C_Result parameter in the case of a retry

Scenario	Receiver's <C_RESULT> parameter value and behaviour	Sender's <C_RESULT> parameter value and behaviour
No error	C_OK: Send FC_ACK and continue reception	C_OK: Receive FC_ACK and continue transmission
SN-Error – C_RetriesMax not exceeded	Send FC_ACK_RET and continue reception	Receive FC_ACK_RET and continue transmission
SN-Error – C_RetriesMax exceeded	C_WRONG_SN: Send FC_ABT and stop reception	C_ERROR Transmission is aborted once FC_ABT is received

7.5.7.2.3.5 C_RetriesMax

The purpose of the retry counter C_RetriesMax is to avoid communication sender and receiver nodes being potentially hooked-up in the case of a fault condition whereby the latter could be that the receiver requests retries continuously. This parameter is local to communication peers and is not transmitted, and is hence not part of the FC protocol data unit.

— The C_RetriesMax counter shall indicate how many retries can be requested by the receiver during an ongoing communication.

— The C_RetriesMax counter upper limit shall be user defined at system generation time and can be modified via the ChangeParameter service.

— The C_RetriesMax counter shall only be used on the receiving network entity during message reception.

— The C_RetriesMax counter shall be set to its initial value any time a new communication starts or a block of data has been successfully transmitted.

7.5.7.3 FS error handling

If an FC C_PDU message is received with an invalid (reserved) FS parameter value, the message transmission shall be aborted, and the communication layer shall make a C_Data.confirm service call with the parameter <C_Result> = C_INVALID_FS to the upper layer.

7.5.7.4 BC error handling

If the receiving entity has correctly received a ConsecutiveFrame/LastFrame that has been transmitted with a smaller separation time as calculated on BC, then the receiver shall accept this frame and continue reception.

7.5.7.5 BfS error handling

No error handling is applicable for BfS.

7.5.7.6 BP error handling

The BP error handling only applies for a message transmitted with acknowledged and retry logic enabled. See Table 30.

Table 30 — BP error handling

BP error	Sender behaviour	Receiver behaviour
BP points to a succeeding buffer segment (not yet transmitted)	Generate a C_Data.confirm with C_Result = C_WRONG_BP and abort the transmission.	A CF timeout occurs after the sender has aborted the transmission and no further CF is received.

7.5.7.7 Unexpected arrival of C_PDU

An unexpected C_PDU is defined as one that has been received by a node outside the expected order of C_PDUs. It could be a C_PDU defined by this part of ISO 10681 (STF C_PDU, CF C_PDU, LF C_PDU or FC C_PDU) that is received out of the normal expected order, or it could be an unknown C_PDU that cannot be interpreted by the definitions given in this document.

As a general rule, the arrival of an unexpected C_PDU from any node shall be handled according to 7.7.

7.5.7.8 Wait frame error handling

If the receiver has

— transmitted MAX_WFT (see 6.3.8) “flow control wait communication layer protocol data units” (FC C_PDU WT) in a row, and

— following this, the receiver cannot meet the performance requirement for the transmission of a “flow control continue to send communication layer protocol data unit” (FC C_PDU CTS),

then the receiver side shall abort the message reception and issue a C_Data.indication with <C_Result> set to C_WFT_OVRN to the upper layer.

The sender of the message is informed about the aborted message reception via a C_Data.confirm with <C_Result> set to C_TIMEOUT_Bs (because of the missing FlowControl C_PDU from the receiver a C_Bs timeout occurs in the sender).

7.6 Communication layer timing

7.6.1 Timing parameters

Figure 15 shows the communication layer timing parameters, while Table 31 defines the communication layer timing parameter values and their corresponding start and end positions based on the data link layer services.

Performance requirement values are the binding communication requirements to be met by each communication peer in order to be compliant with the specification. A certain application may define specific performance requirements within the ranges defined in Table 31.

Timeout values are defined to be higher than the values for the performance requirements 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 in FlexRay dynamic segment). Specified timeout values shall be treated as the lower limit for any given implementation. The real timeout shall occur no later than at the specified timeout value + 50 %.

The communication layer shall issue an appropriate service primitive to the communication layer service user upon detection of an error condition as defined in Table 32.

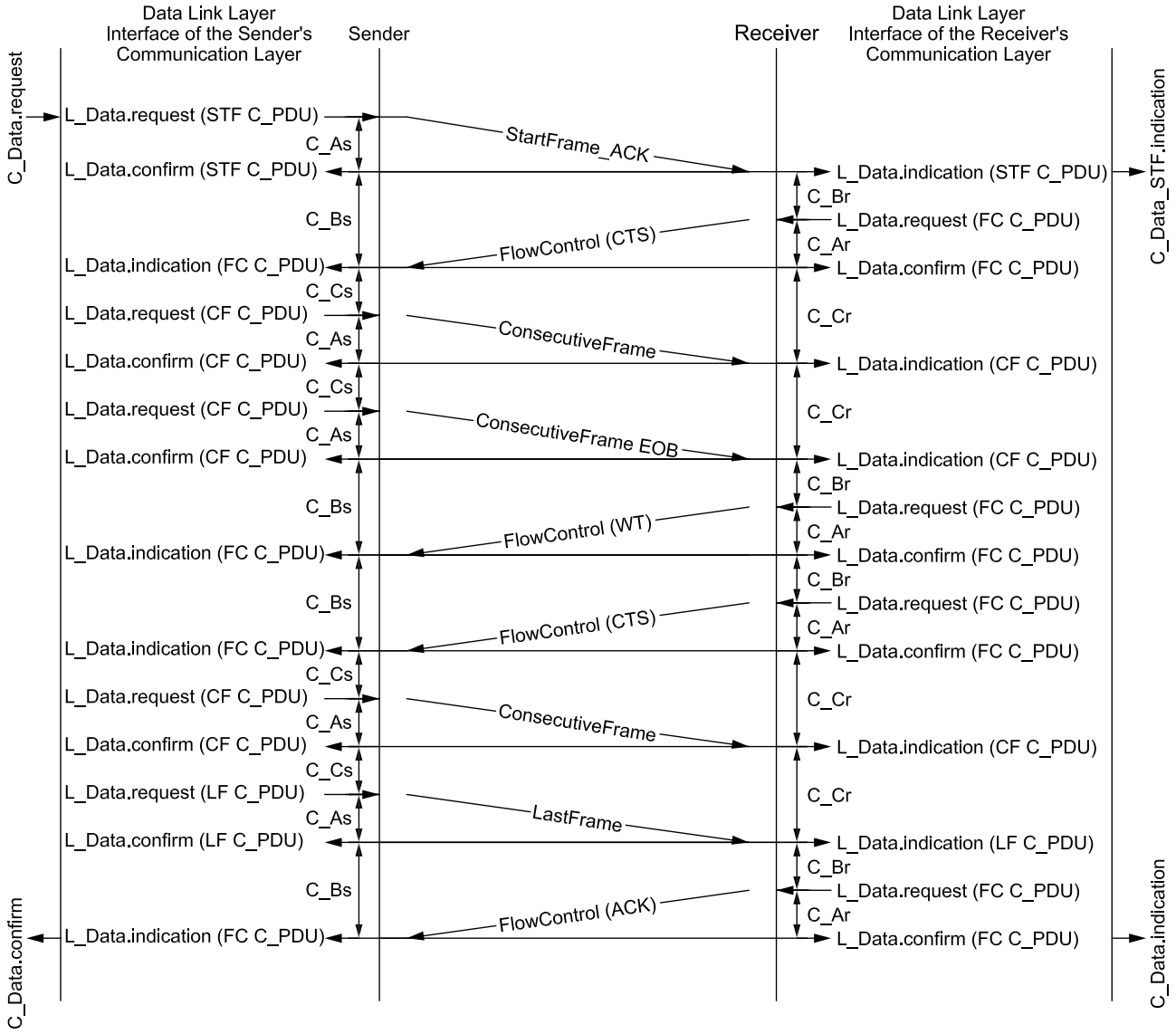


Figure 15 — Placement of communication layer timing parameters

Table 31 —Communication layer timing parameter values

Timing parameter	Description	Data Link Layer services		Timeout	Performance requirement
		Start	End	ms	ms
C_As	Time for transmission of the FlexRay frame (any C_PDU) on the sender side.	L_Data.request	L_Data.confirm	See Formula (1)	N/A
C_Ar	Time for transmission of the FlexRay frame (any C_PDU) on the receiver side	L_Data.request	L_Data.confirm	See Formula (1)	N/A
C_Bs	Time until reception of the next FlowControl C_PDU.	L_Data.confirm(STF), L_Data.confirm(CFx), L_Data.indication(FC), L_Data.confirm(LF)	L_Data.indication (FC)	See Formula (1)	N/A
C_Br	Time until transmission of the next FlowControl C_PDU	L_Data.indication (STF), L_Data.confirm (FC), L_Data.indication (CFx), L_Data.indication (LF)	L_Data.request (FC)	N/A	See Formula (2)
C_Cs	Time until transmission of the next ConsecutiveFrame C_PDU/LastFrame C_PDU	L_Data.confirm (CFx)	L_Data.request (CFx), L_Data.request (LF)	N/A	See Formula (3)
		L_Data.indication (FC)	L_Data.request (CFx), L_Data.request (LF)	N/A	See Formula (4)
C_Cr	Time until reception of the next ConsecutiveFrame C_PDU	L_Data.confirm (FC), L_Data.indication (CFx)	L_Data.indication (CFx), L_Data.indication (LF)	See Formula (1)	N/A

CFx: Represents either CF1, CF2 or CF_EOB

s … is the sender of the message

r … is the receiver of the message

— Formula (1):
Timeout = CycleTime * Factor1

— Formula (2):
Performance Requirement = CycleTime * Factor2

NOTE If an FC C_PDU with a FS equal to WAIT is transmitted, then its transmission shall be delayed to the latest point of the performance requirement C_Br.

— Formula (3):
Performance Requirement = max[CycleTime * Factor2; CycleTime * (SC + 1)]

— Formula (4):
Performance Requirement = CycleTime * Factor2

Furthermore the following conditions apply.

— Factor1 > Factor2 : Both factors are architecture dependent and consider e.g the displacement probability if the dynamic segment is used.

— Factor1 > (SC + 1): SC as defined in 7.5.5.4 applies.

7.6.2 Communication layer timeouts

Table 32 defines the cause and action in a communication layer timeout.

Table 32 — Communication layer timeout error handling

Timeout	Cause	Action
C_As	Any C_PDU not transmitted in time on the sender side.	Abort message transmission and issue C_Data.confirm with <C_Result> = C_TIMEOUT_A
C_Ar	Any C_PDU not transmitted in time on the receiver side.	Abort message reception and issue C_Data.indication with <C_Result> = C_TIMEOUT_A
C_Bs	FlowControl C_PDU not received (lost, overwritten) on the sender side or preceding StartFrame C_PDU or ConsecutiveFrame C_PDU not received (lost, overwritten) on the receiver side.	Abort message transmission and issue C_Data.confirm with <C_Result> = C_TIMEOUT_Bs
C_Cr	ConsecutiveFrame or Last Frame C_PDU not received (lost, overwritten) on the receiver side or preceding FC C_PDU not received (lost, overwritten) on the sender side.	Abort message reception and issue C_Data.indication with <C_Result> = C_TIMEOUT_Cr

7.7 Interleaving of messages

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

Table 33 determines the behaviour of the communication layer when an unexpected arrival of a C_PDU occurs in either a half- or full-duplex environment.

Table 33 — Handling of unexpected arrival of C_PDU

CL status	Reception of					
	STF C_PDU unsegmented	STF C_PDU segmented	CF C_PDU	FC C_PDU	LF C_PDU	Unknown C_PDU
Segmented transmit in progress	Full-duplex: If a reception is in progress, see the corresponding cell below in this table. Otherwise, process the STF C_PDU as the start of a new reception.	Full-duplex: If a reception is in progress, see the corresponding cell below in this table. Otherwise, process the STF C_PDU as the start of a new reception.	Full-duplex: If a reception is in progress, see the corresponding cell below in this table.	If awaited, process the FC C_PDU. Otherwise, ignore it.	Full-duplex: If a reception is in progress, see the corresponding cell below in this table.	Ignore
	Half-duplex: ignore	Half-duplex: ignore	Half-duplex: ignore		Half-duplex: ignore	
Segmented receive in progress	Terminate the current reception, report a C_Data.indication, with <C_Result> set to C_UNEXP_PDU, to the upper layer, and process the STF C_PDU as the start of a new reception.	Terminate the current reception, report a C_Data.indication, with <C_Result> set to C_UNEXP_PDU, to the upper layer, and process the STF C_PDU as the start of a new reception.	If awaited, process the CF C_PDU in the on-going reception and perform the required checks (e.g. SN in right order). Otherwise, ignore it.	Full-duplex: If a transmission is in progress, see the corresponding cell above in this table.	If awaited, process the LF C_PDU in the on-going reception and perform the required checks. Otherwise, ignore it.	Ignore
				Half-duplex: Ignore		
Idle[a]	Process the STF C_PDU as the start of a new reception.	Process the STF C_PDU as the start of a new reception.	Ignore	Ignore	Ignore	Ignore

[a] Neither a segmented transmission nor an unsegmented transmission is in progress.

8 Data link layer usage

8.1 General

This data link layer usage clause describes the interface between the communication layer and the FlexRay data link layer (DLL).

8.2 Rules

A single communication (segmented or unsegmented transmission) between nodes takes place completely in either the static segment or the dynamic segment.

All C_PDUs of a multiframe transmission are transmitted on one FlexRay channel (channel A or channel B) or transmitted on both FlexRay channels (if only the static segment is used). It is required to ensure temporal ascending order of C_PDUs (sequential order of ConsecutiveFrames)[3)].

Two different methods for C_PDU into L_PDU placements are applicable.

— A single C_PDU is mapped to a single L_PDU

3) Basically required for the retry mechanism handling. Furthermore, this avoids re-ordering of PDUs on the receiver side.

— This allows the usage of the preamble indication (address filtering) if the L_PDU is placed in the beginning of a FlexRay frame.

— This allows the dynamic payload length to be supported.

— Multiple C_PDUs with a defined fixed length are mapped to a single L_PDU.

— Only C_PDUs (no other N_PDUs) shall be mapped into single L_PDU.

If redundancy is required (simultaneous communication on both FlexRay channels), the data link layer needs to ensure synchronized transmission.

A receiving node must receive all L_PDUs of a potential transmitting entity. For the reception of L_PDUs, a node can choose between the following two methods:

a) Hardware address filtering (payload preamble indication bit utilized): The communication controller of the node filters the address information and ignores all L_PDUs which are not addressed to this node.

b) Software address filtering (payload preamble indication bit not utilized): The node must filter the address information by software and must ignore all L_PDUs which are not addressed to this node.

Figures 16 to 20 provide examples for the rules given above to clarify what is allowed and what is not allowed.

NOTE The examples given assume that reaction on received C_PDUs is done by the receiving node within the same cycle this C_PDU is received, which is probably a rare case, but simplifies the figures for explanation.

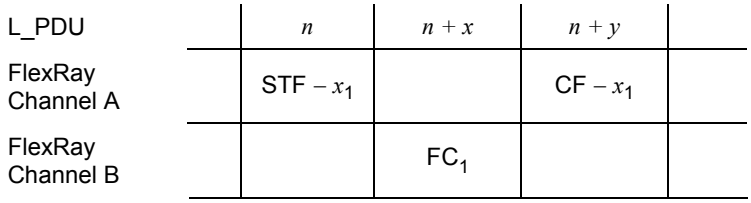


Figure 16 — Example A: Single transmission on different channels

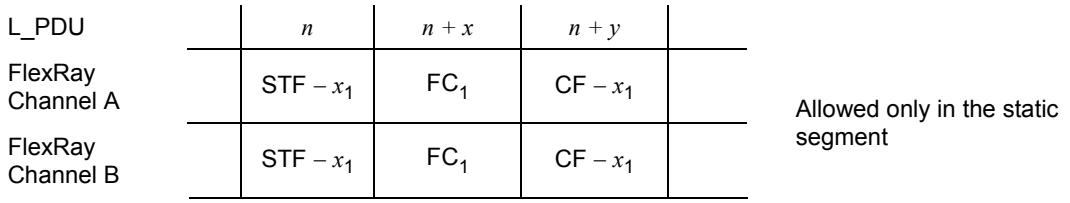


Figure 17 — Example B: Single redundant segmented transmission on different channels (static segment only)

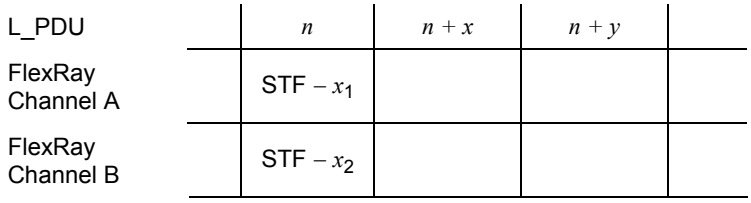


Figure 18 — Example C: Two different non-redundant unsegmented transmissions on different channels

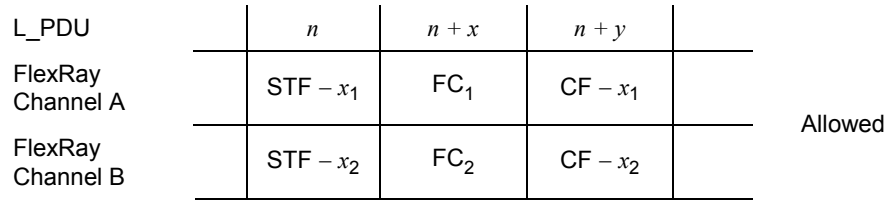


Figure 19 — Example D: Two different non-redundant segmented transmissions on different channels

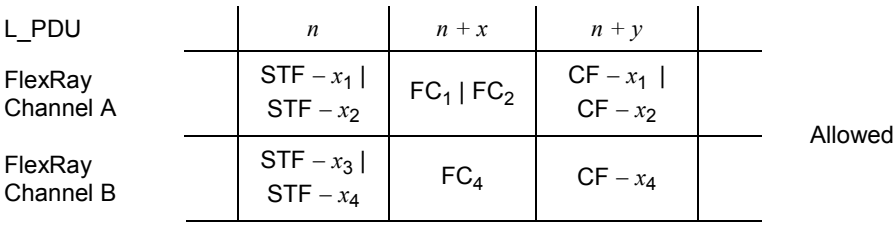


Figure 20 — Example E: Three different non-redundant segmented transmissions and one unsegmented transmission on different channels

8.3 Data flow between CL and DLL

8.3.1 Transmit data flow

The data flow mechanism between the CL and the DL uses the following items to perform the required functionality.

— Queues (FIFO)

Queues are resources used to handle concurrent communication layer transmissions. Each queue holds C_PDUs of a single communication layer message. The CL reserves a queue for a single transmission and frees it after the confirmation of the transmission of the final C_PDU of that particular message.

— Filling-Level

Each queue is controlled by a Filling-Level that provides the following features:

— it is incremented by the CL whenever the transmission of a C_PDU is requested by placing it in the queue;

— it is decremented by the CL whenever a C_PDU is placed in a L_PDU for transmission (suitable L_PDU found).

This handling prevents queue overflow and is used to identify confirmation of the final C_PDU. After confirmation of the final C_PDU, the queue will be released by the CL and is free for further use.

— Tx-Pending-Counter

Each queue is controlled by Tx-Pending-Counter that provides the following features:

— it is incremented by the DLL whenever a C_PDU is placed in a L_PDU for transmission (suitable L_PDU found);

— it is decremented by the DLL whenever a transmission of a C_PDU has been confirmed

For details about the handling, see Figure 21 and following description.

— DLL Scheduler:

The DLL scheduler is used to fill the L_PDUs which are reserved for CL transmissions with C_PDUs out of the active queues and is executed several times during a FlexRay cycle. It uses a round-robin mechanism to ensure that the L_PDUs are filled in a coordinated manner that guarantees that all active transmissions are served concurrently. A transmission started on a particular FlexRay channel has to continue on that channel until its end. The communication layer defines on which channel the transmission shall take place:

— FlexRay channel A;

— FlexRay channel B; or

— FlexRay channel A and B redundantly.

The determination of the FlexRay channel and segment to be used is done by the communication layer via evaluation of the address information (C_TA).

Figure 21 graphically depicts an example for the data flow and interfacing between the communication layer and the data link layer.

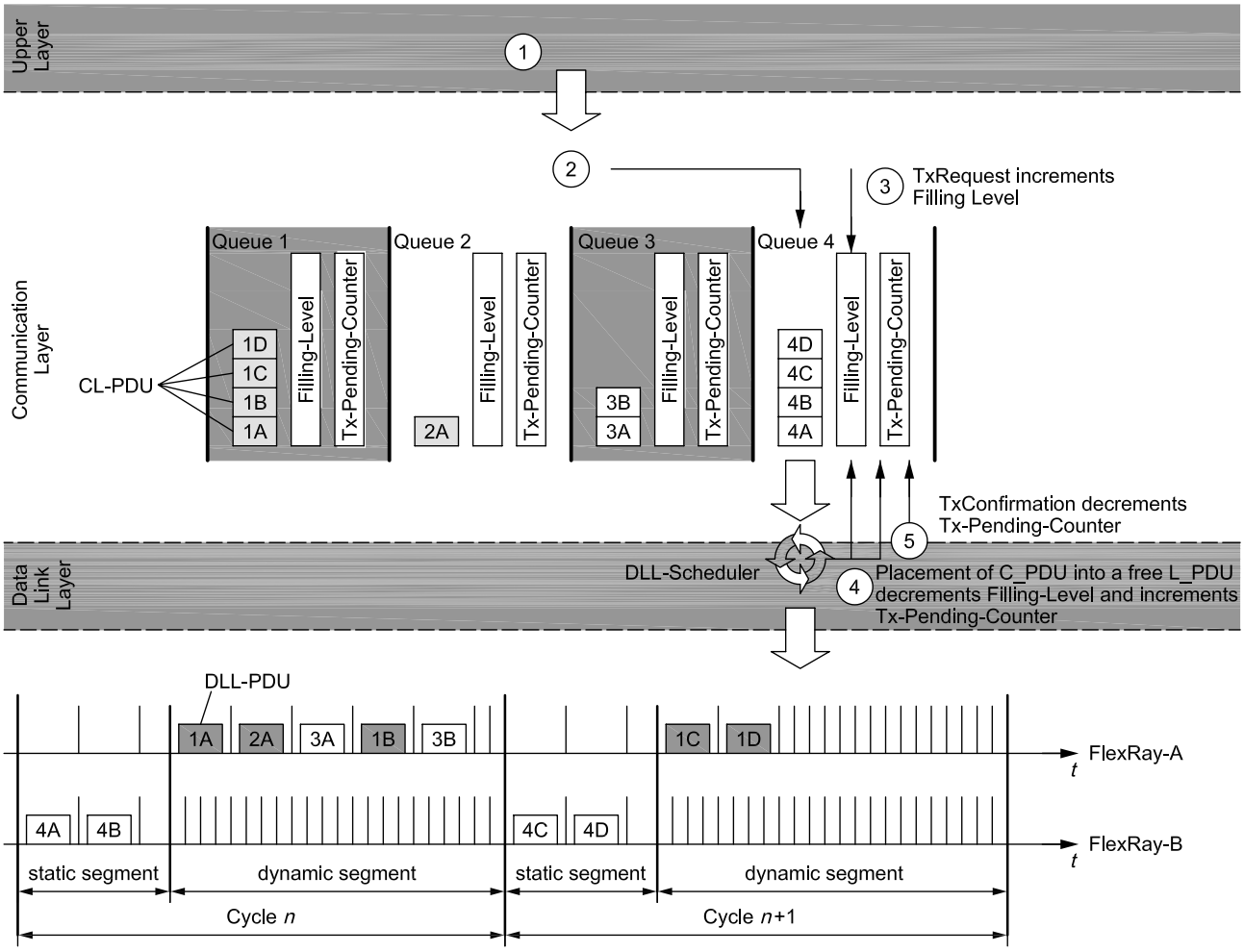


Figure 21 — Communication layer PDU distribution for transmission

Handling description (referencing the numbers given in Figure 21):

1) The communication layer receives a transmit request from the upper layer.

2) The CL reserves a transmit queue for the complete transmission of this particular message (e.g. Queue 4). This reservation includes the allocation of the FlexRay channel(s) to be used (see definitions above).

3) The CL starts filling the queue with C_PDUs (e.g. 4A, 4B, 4C, etc.) as long as space in the queue is available according to the Filling-Level. With each inserted C_PDU, the CL increments the Filling-Level by 1.

4) The DLL scheduler retrieves the C_PDUs from each queue in ascending order. The following conditions have to be satisfied in order to retrieve a C_PDU from an individual queue and place it in the next free L_PDU.

   — Filling-Level of the individual queue must be greater than zero.

   — Tx-Pending-Counter of the individual queue must be equal to zero at the start of a new FlexRay cycle.

   — A suitable L_PDU must be available for transmission.

   A suitable L_PDU fulfils the following conditions.

   — The FlexRay channel(s) as given in the queue attributes must be met (see above).

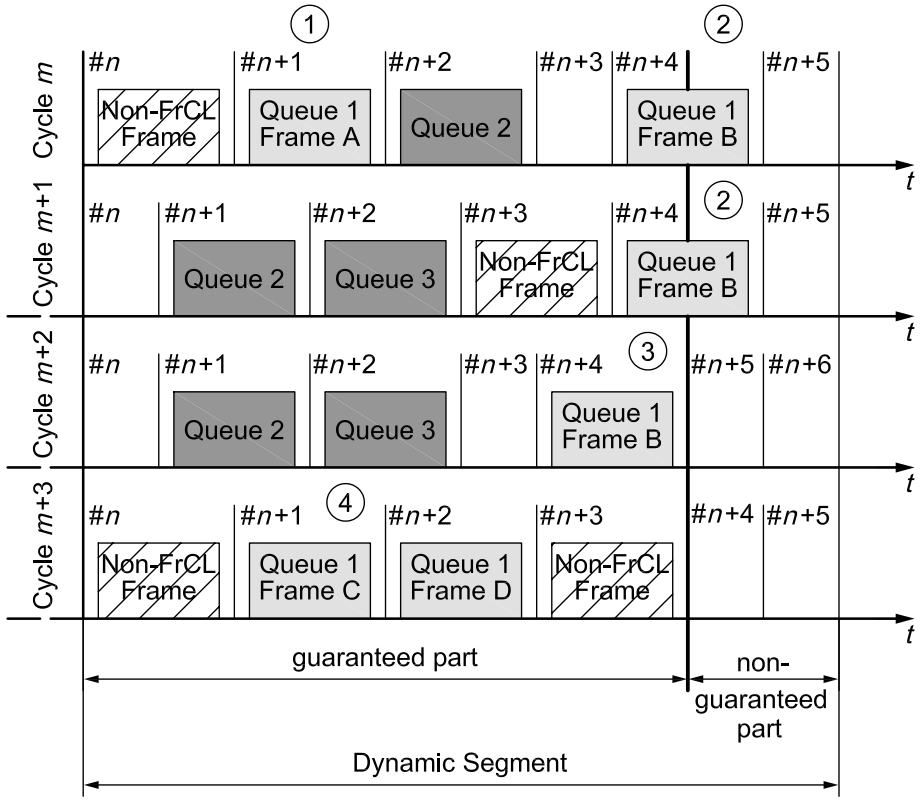
   — All subsequent C_PDUs retrieved from the same queue must be transmitted on the same preselected FlexRay channel.

   — L_PDU is not occupied by another queue.

   If a suitable L_PDU has been found then the Tx-Pending-Counter is incremented by 1 and the Filling-Level is decremented by 1. If no suitable condition is found the DLL scheduler switches to the next queue and evaluates the possibility for a transmission again.

5) The DLL scheduler generates a transmit confirmation for each transmitted C_PDU. Each confirmation decrements the Tx-Pending-Counter (by 1) of the particular queue where the confirmed C_PDU belongs to. A transmission might be executed in a following FlexRay cycle because the latest point of transmission (pLatestTx) has elapsed. Therefore, the DLL scheduler needs to postpone the transmission until the Tx-Pending-Counter of the affected queue is equal to zero.

Figure 22 shows a message transfer example where three different L_PDU's are assigned within the dynamic segment. The picture considers a multi frame message transfer (consecutive frames only):



FrCL = FlexRay Communication Layer

Figure 22 — Example for message transfer mechanism on CL and DLL Interface (API)

For the example given, the L_PDUs n + 1, n + 2 and n + 4 are assigned for CL communication. The repetition for the L_PDUs is equal to 1.

1) The first consecutive frame (CF) is transmitted within the guaranteed part of the dynamic segment

2) The second CF won't be transmitted within the same cycle (m and m + 1) because the assigned slot is within the non-guaranteed part and 'pLatestTx' has been passed already

3) The second CF will be transmitted within cycle m + 2.

4) After a successful transmission of the second CF (positive confirmation received), the transmission of the third CF is started in cycle m + 3.

8.3.2 Receive data flow

Figure 23 depicts the receive data flow.

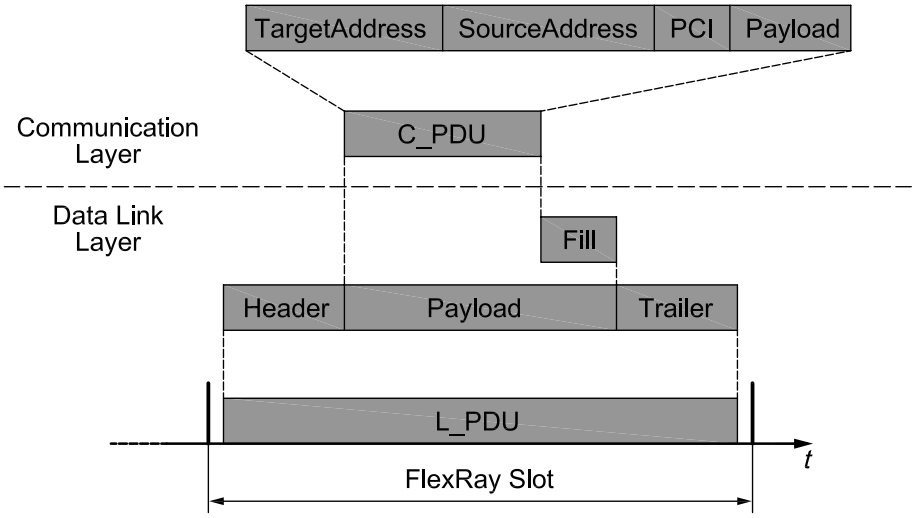


Figure 23 — Communication layer PDU distribution for reception

— The DLL receives any FlexRay frame configured for communication layer handling (L_PDU). The header and trailer are stripped off and the payload data (C_PDU) is forwarded to the communication layer.

— The communication layer interprets the address information embedded in the received C_PDU in order to identify whether the node is addressed. If the address matches, the C_PDU is processed.

8.4 Data link layer interface services

8.4.1 L_Data.request

The request primitive requests transmission of <Data> that shall be mapped within specific attributes of the data link protocol data unit selected by means of <Identifier>, <FRChannel>, <FRSegment> and <Length>

```
L_Data.request   (
                 <Identifier>,
                 <FRChannel>,
                 <FRSegment>,
                 <ReqLength>,
                 <ReqData>
                 )
```

8.4.2 L_Data.indication

The service primitive indicates data link layer event to the upper layer and delivers <Data> identified by a <Length>. Address information are part of the <Data> which will be evaluated by the upper layer. Any <Identifier> information is not necessary because the address information (TA, SA, etc.) is part of the payload information. Furthermore, no PDU filtering is part of the communication layer which would require an <Identifier> information.

```
L_Data.indication  (
                   <IndLength>,
                   <IndData>
                   )
```

8.4.3 L_Data.confirm

The service primitive confirms the completion of an L_Data.request service for a specific <Identifier>.

L_Data.confirm (
<Identifier>,
<L_Result>
)

8.5 Data link layer service parameters

The following data link layer service parameters are defined:

<Identifier> Identifier which is assigned to a transmission queue of the CL (API specific value such as a handle).

Type: unsigned 8 bits

Range: 0 to FF hex

<FRChannel> This parameter identifies the FlexRay channel(s) to be used for the transmission of the C_PDU:

Type: enumeration

Range: channel_A, channel_B, channel_AB

NOTE The communication layer evaluates the address information (C_TA) in order to select the FlexRay channel.

<FRSegment> This parameter identifies the FlexRay segment to be used for the transmission of the C_PDU:

Type: enumeration

Range: static, dynamic

NOTE The communication layer evaluates the address information (C_TA) in order to select the FlexRay segment.

<ReqLength> Length of the <Data> field, size of C_PDU to be transmitted (see Figure 23).

Type: unsigned 8 bits

Range: 5 to FE hex

<ReqData> This parameter includes all data of the C_PDU to be transmitted.

Type: string of bytes with length <ReqLength>

<IndLength> Length of the <Data> field, size of C_PDU received also including potential fill bytes.

Type: unsigned 8 bits

Range: 6 to FE hex

<IndData> This parameter includes all data received, including the C_PDU and potential fill-bytes.

Type: string of bytes with length <IndLength>

<L_Result> This parameter contains the status relating the execution of the L_Data.request service primitive.

Type: enumeration

Range: L_OK, WRONG_FR_CHANNEL, L_WRONG_FR_SEGMENT, WRONG_LENGTH

— L_OK

This value means that the service execution has completed successfully.

— L_WRONG_FR_CHANNEL

This value is issued to the service user to indicate that the service did not execute due to an invalid <FRChannel> parameter.

— L_WRONG_FR_SEGMENT

This value is issued to the service user to indicate that the service did not execute due to an invalid <FRSegment> parameter.

— L_WRONG_LENGTH

This value is issued to the service user to indicate that the service did not execute due to an invalid <Length> parameter.

— L_NOT_OK

This value is issued to the service user if an immediate transmission is conducted with failure by the DLL, e.g. the FlexRay-CC refuses access to the Tx-object.

8.6 Mapping of C_PDU fields

Table 34 specifies the mapping of the communication layer C_PDUs onto the FlexRay data link layer L_PDU payload field.

Table 34 — Mapping of C_PDU into the L_PDU payload field

Name	**L_PDU payload field**										
Byte position	B1-2	B3-4	B5	B6	B7	B8	B9	B10	B11	…	Bn
StartFrame (STF)	C_TA	C_SA	C_PCI				D1	D2	D3	…	Dm
ConsecutiveFrame_x (CFx[a])	C_TA	C_SA	C_PCI		D1	D2	D3	D4	D5	…	Dm
FlowControl (FC_CTS)	C_TA	C_SA	C_PCI								
FlowControl (FC_ACK/RET)	C_TA	C_SA	C_PCI								
FlowControl (FC_ABT, FC_WT)	C_TA	C_SA	C_PCI								
LastFrame (LF)	C_TA	C_SA	C_PCI				D1	D2	D3	…	Dm

[a] x = 1 or 2, representing ConsecutiveFrame_1, ConsecutiveFrame_2 and ConsecutiveFrame_EOB

NOTE Grayed out cells are not utilized for data information.

8.7 Hardware Address filtering within dynamic segment

The Payload Preamble Indicator (1 Bit) in combination with the MessageIdentifier might (optional feature) be used for hardware acceptance filtering within the dynamic segment. It can be used based on current C_PDU mapping onto FlexRay L_PDU payload field (see Table 34).

For communication within the static segment, the Payload Preamble Indicator shall always be set to zero.

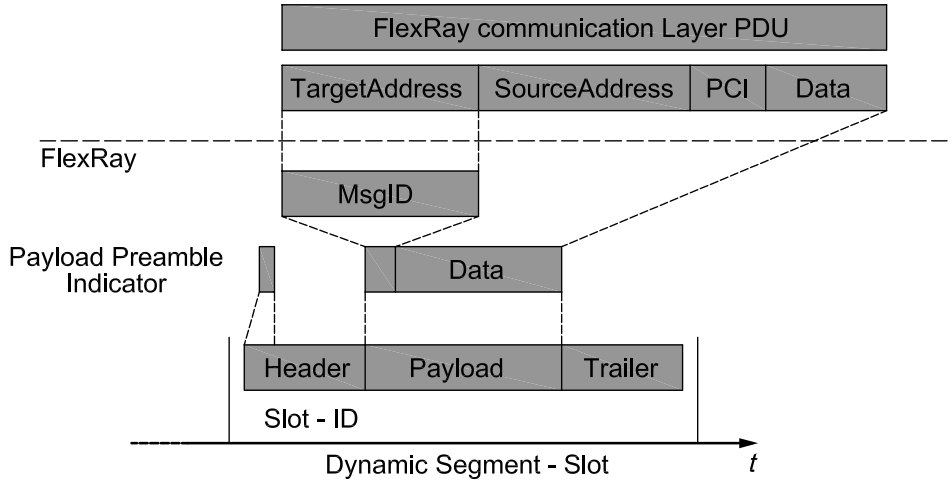


Figure 24 — Target address mapping for dynamic segment

NOTE For simplicity, potential fill Bytes on the FlexRay Data Link Layer are not shown in the Figure 24.

8.8 Multiple C_PDU to single L_PDU mapping

Figure 25 graphically depicts a concept utilizing the potential placement of multiple C_PDUs into a single L_PDU as stated in 8.2.

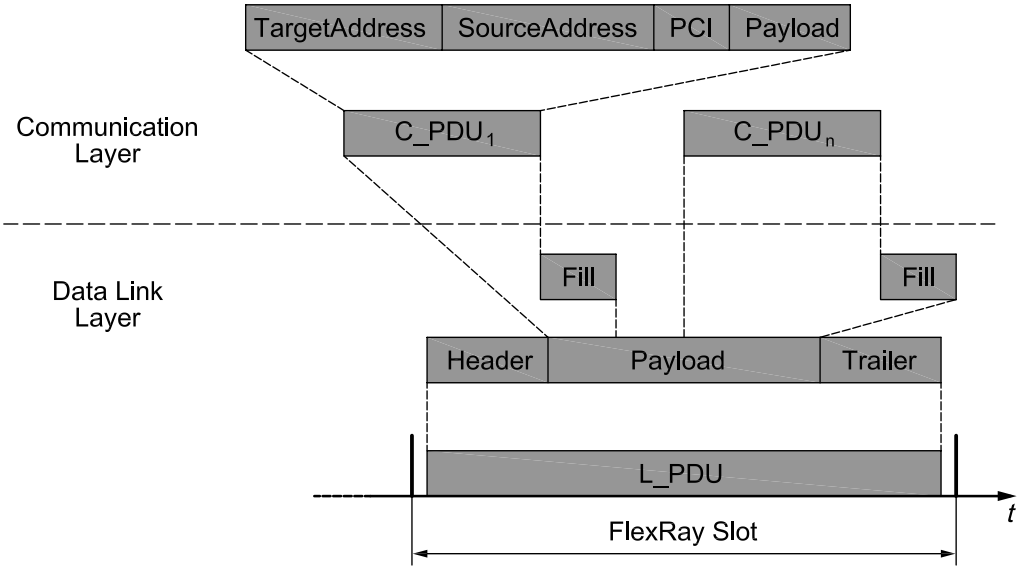


Figure 25 — Multiple C_PDU to single L_PDU mapping

Annex A
(informative)

Implementation examples

A.1 Scope

This annex shows a possible implementation of an API which considers buffer constraints in the sender and the receiver. It is assumed that there is at least the memory available for a single C_PDU to be transmitted in a SingleFrame; therefore, this document focuses on the following cases only.

— Case #1: Message Length < TxBufferSize, MessageLength < Rx Buffer

— Case #2: Message Length > TxBufferSize, MessageLength < Rx Buffer

— Case #3: Message Length < TxBufferSize, MessageLength > Rx Buffer

— Case #4: Message Length > TxBufferSize, MessageLength > Rx Buffer

A.2 Case analysis

A.2.1 Case #1: Message Length < TxBufferSize, MessageLength < Rx Buffer

In the following case, the message to be transmitted fits into the transmit buffer and receive buffer completely. Figure A.1 depicts this.

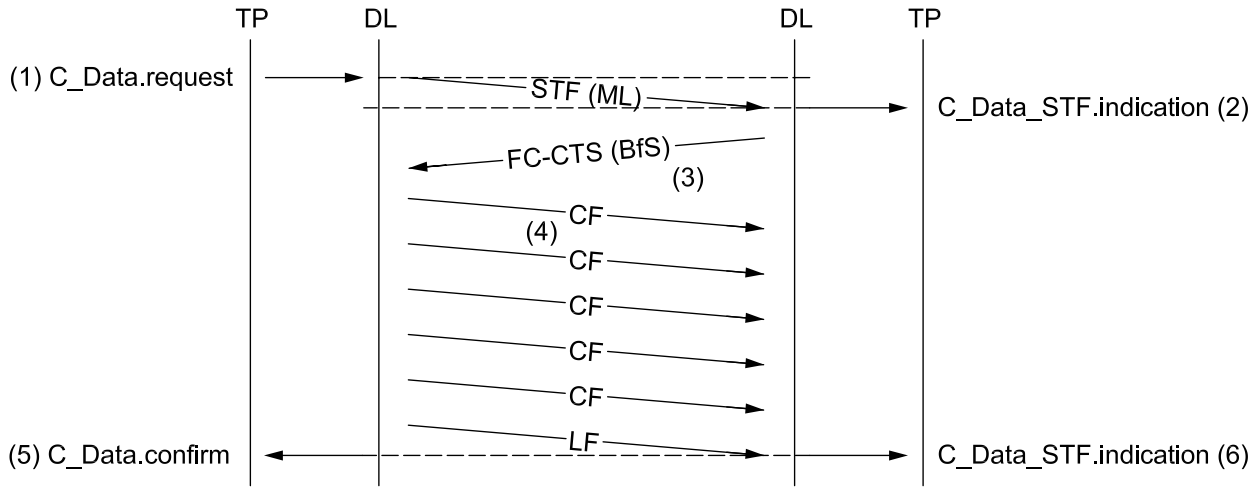


Figure A.1 — ML<TxBufferSize, ML<RxBufferSize

1) The sender requests the transmission of a message with known MessageLength (ML) set to the total length of the message and <ActualLength> = 0 via C_Data.request. The caller knows that its communication layer buffer can hold the complete message. Therefore, the caller hands over the complete message at once.

2) The StartFrame transmitted by the sender of the message is received by the receiver and indicated to the upper layer via C_Data_STF.indication.

3) The receiver reports its buffer capabilities back to the sender in the FlowControl frame via the BfS parameter. For the given case, the complete message fits into the buffer of the receiver (BfS ⩾ ML).

4) The sender starts the block following the FlowControl frame with a Consecutive Frame.

5) After the transmission of the LastFrame (LF), the completion is indicated in the sender via C_Data.confirm (assumed it is an unacknowledged transmission).

6) The completion of the reception is indicated in the receiver via C_Data.indication. The LastFrame is potentially not completely filled.

A.2.2 Case #2: Message Length > TxBufferSize, MessageLength < Rx Buffer

In the following case the message to be transmitted does not fit completely into the transmit buffer, but it fits completely into the receive buffer. Figure A.2 depicts this.

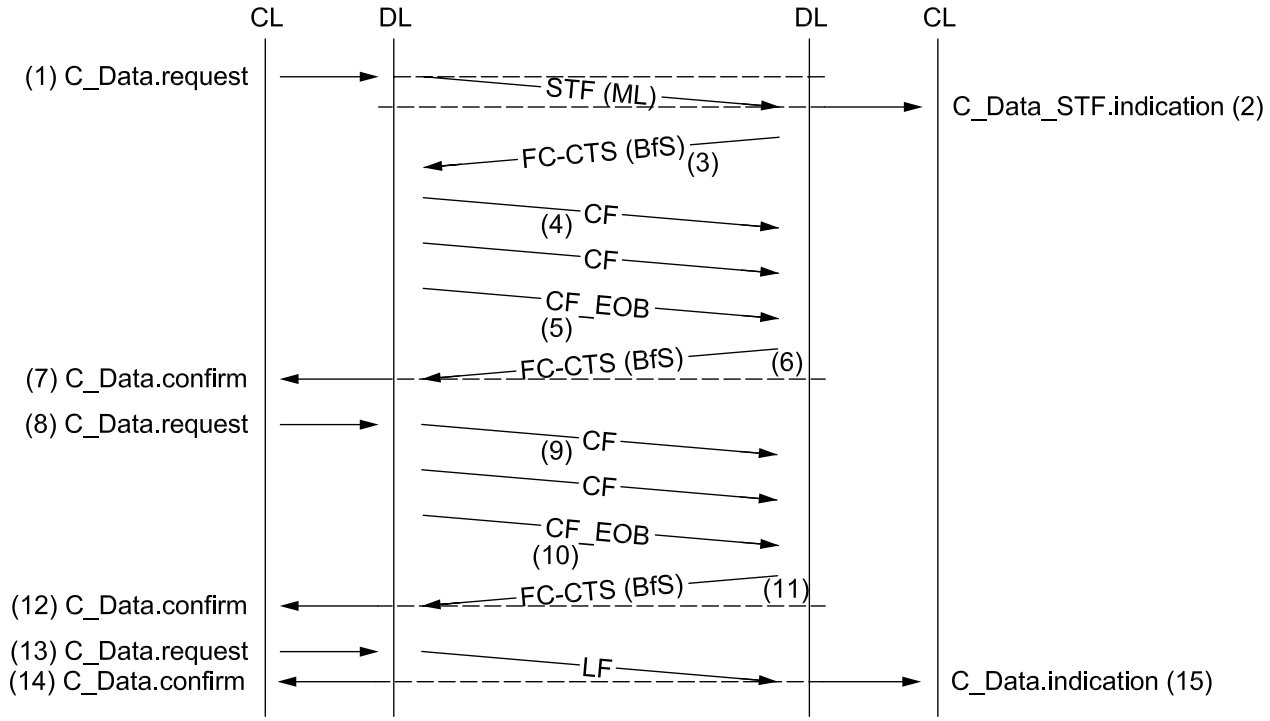


Figure A.2 — ML>TxBufferSize, ML<RxBufferSize

1) The sender requests the transmission of a message with MessageLength (ML) set to the total length of the message via C_Data.req but hands over only the amount of data that fits into the sender buffer (ActualLength <> 0). The StartFrame transmitted indicates the complete message via ML.

2) The StartFrame of the message is received by the receiver and indicated to the upper layer via C_Data_STF.indication.

3) The receiver reports its buffer capabilities back to the sender in the FlowControl frame via the BfS parameter. For the given case, the complete message fits into the buffer of the receiver (BfS ⩾ ML).

4) The sender starts the block following the FlowControl frame with a ConsecutiveFrame.

5) The last frame of the block is a ConsecutiveFrame End of Block (CF_EOB) and indicates that an acknowledgement of that segment is required from the receiver.

6) The receiver acknowledges the reception of the ConsecutiveFrame End of Block (CF_EOB) via a Flow Control. This acknowledgment is used in the sender to adjust its buffer pointers accordingly and to throw away the data of the completely transmitted block.

7) The communication layer indicates to the upper layer to feed in further data for this message via C_Data.con with C_Result = C_MOREDATA.

8) Further data are handed over using the C_Data.req service primitive, taking into account the transmit buffer size as in the first block. The sender now continues to transmit the data.

9) The sender starts the next block following the FlowControl frame with a ConsecutiveFrame.

10) The CF_EOB frame indicates that an acknowledgement of that segment is required from the receiver.

11) The receiver acknowledges the reception. The acknowledgment is used in the sender to adjust its buffer pointers accordingly and to throw away the data of the completely transmitted block.

12) The communication layer indicates to the upper layer to feed in further data for this message via C_Data.con with C_Result = C_MOREDATA.

13) The remaining message data are handed over using the C_Data.req service primitive. The sender now continues to transmit the data within the Last Frame.

14) Now the complete message is transmitted. After the transmission of the Last Frame (assumed it is an unacknowledged transmission with a known message length) the completion is indicated in the sender via C_Data.con with C_Result = C_OK.

15) The completion of the reception is indicated in the receiver via C_Data.ind.

A.2.3 Case #3: Message Length < TxBufferSize, MessageLength > Rx Buffer

In this case, the message to be transmitted fits completely into the transmit buffer, but it does not fit completely into the receive buffer. Figure A.3 depicts this.

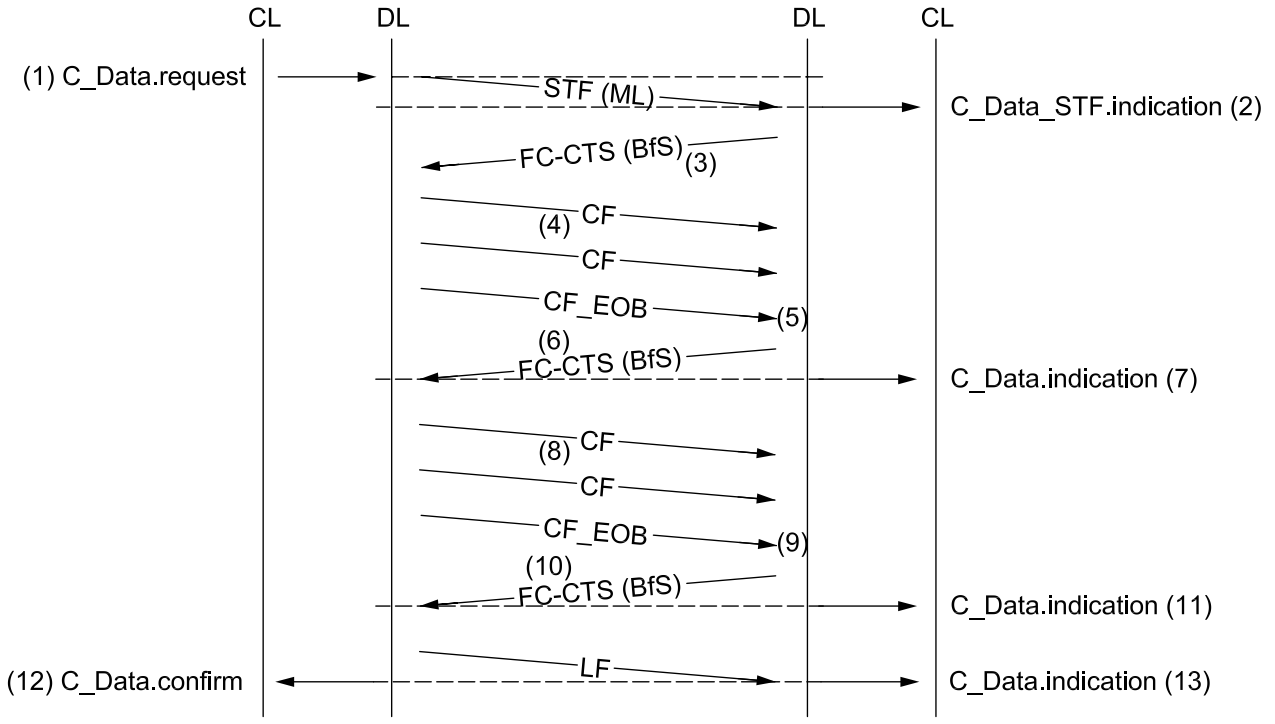


Figure A.3 — ML<TxBufferSize, ML>RxBufferSize

1) The sender requests the transmission of a message with MessageLength (ML) via C_Data.req and hands over the complete message, because it fits into its communication layer (ActualLength = 0).

2) The StartFrame of the message is received by the receiver and indicated to the upper layer via C_Data_STF.indication.

3) The receiver reports its buffer capabilities back to the sender in the FlowControl frame via the BfS parameter. For the given case, only part of the message fits into the buffer of the receiver (BfS < ML).

4) The sender starts the block following the FlowControl frame with a ConsecutiveFrame.

5) The last frame of the block (CF_EOB) indicates that an acknowledgement of that block is required from the receiver, because the sender uses the reported BfS information based on the receiver buffer capabilities to perform the block acknowledgement.

6) The receiver acknowledges the reception. There is no adjustment of the buffer in the sender required.

7) In the receiver, the data received up to that point are reported to the upper layer via C_Data.ind with C_Result = C_MOREDATA and the buffer is adjusted, so it can be used to receive the next block.

8) The sender starts the next block following the FlowControl frame with a ConsecutiveFrame.

9) The last frame of the block (CF_EOB) indicates that an acknowledgement of that block is required from the receiver, because the sender uses the reported BfS information based on the receiver buffer capabilities to perform the block acknowledgement.

10) In the receiver, the data received up to that point are reported to the upper layer via C_Data.ind with C_Result = C_MOREDATA and the buffer is adjusted, so it can be used to receive the next block.

11) The receiver acknowledges the reception. There is no adjustment of the buffer in the sender required.

12) Now the complete message is transmitted. After the transmission of the Last Frame (assumed it is an unacknowledged transmission with a known message length), the completion is indicated in the sender via C_Data.con.

13) The completion of the reception is indicated in the receiver via C_Data.ind with C_Result =C_OK.

A.2.4 Case #4: ML > TxBufferSize, ML > Rx Buffer

In this case, the message to be transmitted does not fit completely into the transmit buffer nor the receive buffer. Figure A.4 depicts this.

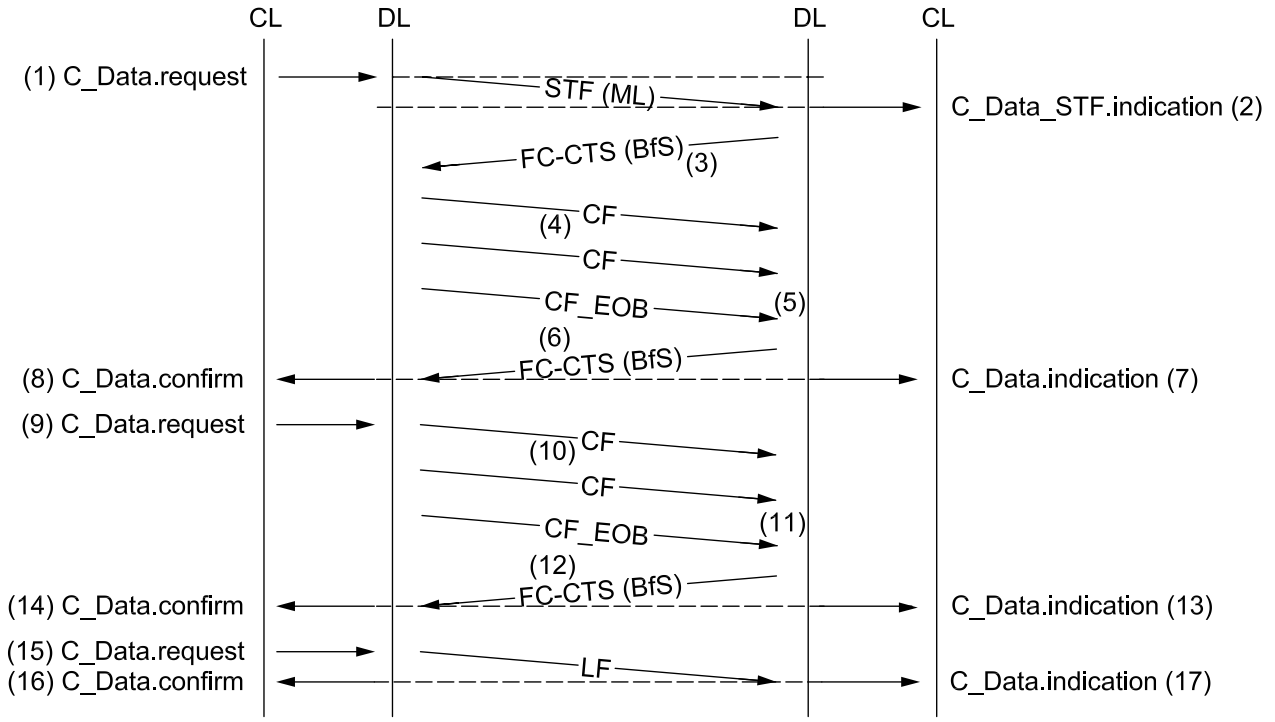


Figure A.4 — ML>TxBufferSize, ML>RxBufferSize

1) The sender requests the transmission of a message with MessageLength (ML) set to the total length of the message via C_Data.req but hands over only the amount of data that fits into the sender buffer (ActualLength <> 0). The StartFrame transmitted indicates the complete message via ML.

2) The StartFrame of the message is received by the receiver and indicated to the upper layer via C_Data_STF.indication.

3) The receiver reports its buffer capabilities back to the sender in the FlowControl frame via the BfS parameter. For the given case, only part of the message fits into the buffer of the receiver (BfS < ML).

4) The sender starts the block following the FlowControl frame with a Consecutive Frame.

5) The last frame of the block (CF_EOB) indicates that an acknowledgement of that block is required from the receiver. This can either be forced by the buffer constraints in the sender or via the reported BfS of the receiver.

6) The receiver acknowledges the reception.

7) In the receiver, the data received up to that point are reported to the upper layer via C_Data.ind with C_Result = C_MOREDATA and the buffer is adjusted, so it can be used to receive the next block.

8) The communication layer in the sender indicates to the upper layer to feed in further data for this message via C_Data.con with C_Result = C_MOREDATA.

9) The upper layer in the sender now provides further data to be transmitted.

10) The sender starts the next block following the FlowControl frame with a ConsecutiveFrame.

11) The last frame of the block (CF_EOB) indicates that an acknowledgement of that segment is required from the receiver. This can either be forced by the buffer constraints in the sender or via the reported BfS of the receiver.

12) The receiver acknowledges the reception.

13) In the receiver, the data received up to that point are reported to the upper layer via C_Data.ind with C_Result = C_MOREDATA and the buffer is adjusted, so it can be used to receive the next block.

14) The communication layer in the sender indicates to the upper layer to feed in further data for this message via C_Data.con with C_Result = C_MOREDATA.

15) The upper layer in the sender now provides further data to be transmitted.

16) Now the complete message is transmitted. After the transmission of the Last Frame (assumed it is an unacknowledged transmission with a known message length), the completion is indicated in the sender via C_Data.con with C_Result = C_OK.

17) The completion of the reception is indicated in the receiver via C_Data.ind with C_Result = C_OK.

Bibliography

[1] AUTOSAR, *Specification of Module FlexRay Transport*

[2] ISO 14229-1 [4], *Road vehicles — Unified diagnostic services (UDS) — Part 1: Specification and requirements*

[3] ISO 14229-2[5], *Road vehicles — Unified diagnostic services (UDS) — Part 2: Session layer services*

[4] ISO 14229-4[5], *Road vehicles — Unified diagnostic services (UDS) — Part 4: Unified diagnostic services on FlexRay implementation (UDSonFR)*

[5] *FlexRay Communications Systems Protocol Specification*

4) Under preparation. (Revision of ISO 14229-1:2006)

5) Under preparation.

ICS 43.040.15

Price based on 64 pages