Annex C: Data Transfer Sublayer (Mandatory)

The Data Transfer Sublayer is responsible for the efficient transfer across the radio link of protocol data units from the Management Sublayer (i.e., M_PDUs) and Channel Access Sublayer (i.e., C_PDUs) in the STANAG 5066 profile. Usually, but not always, this means the error free delivery of C_PDUs to the Channel Access Sublayer. Efficient transmission over the radio channel of protocol data units of the Data Transfer Sublayer (i.e., D_PDUs, which contain the C_PDUs or M_PDUs) is achieved principally by two mechanisms. The first is segmentation of the large C_PDUs into smaller D_PDUs if necessary and the second is the selective repetition (selective ARQ) by the sending node of D_PDUs which were received in error. The other mode (non ARQ) of transmitting involves the segmentation of the large C_PDUs into smaller D_PDUs and combining the received D_PDUs such that the segmented C_PDU can be reconstructed in a "best-effort" attempt.

C.1 Data Transfer Sublayer Service Definition

Depending on the application and service-type requested by higher sublayers, the user service provided by the Data Transfer Sublayer **shall** (1) be either a simple **non ARQ service** - commonly known as broadcast mode - or a reliable **selective ARQ service**, as specified herein.

The Data Transfer Sublayer **shall** (2) provide "sub-modes" for **non ARQ** and reliable **selective ARQ** delivery services, which influence the characteristics of the particular service, as specified below

In addition to the Selective ARQ and Non-ARQ services provided to the upper sublayers, the Data Transfer Sublayer shall provide an Idle Repeat Request service for peer-to-peer communication with the Data Transfer Sublayer of other nodes.

C.1.1 Non-ARQ Service

In the **non ARQ service** error-check bits (i.e., cyclic-redundancy-check or CRC bits) applied to the D_DPU **shall** (1) be used to detect errors, and any D_PDUs that are found to contain transmission errors **shall** (2) be discarded by the data transfer sublayer protocol entity.

A special mode of the non-ARQ service **shall** (3) be available to reconstruct C_PDUs from D_PDUs in error and deliver them to the Channel Access Sublayer.

In the **non ARQ** mode, the following submodes may be specified:

- regular data service.
- expedited data service.
- "in order" delivery of C_PDUs is not guaranteed.
- delivery of complete or error free C_PDUs is not guaranteed.

C.1.2 Selective ARQ Service

The reliable **Selective ARQ service shall** (1) use CRC check bits and flow control procedures, such as requests for retransmission of D_PDUs in which errors have been detected, to provide a reliable data transfer service.

In the **Selective ARQ** service, the following submodes may be specified:

- regular data service.
- expedited data service.
- node and client level delivery confirmation.
- "in-order" or "out-of-order" deliver of C_PDUs to the remote peer network layer.
- Delivery of complete and error free C_PDUs is guaranteed.

C.2 Interface Primitives Exchanged with the Channel Access Sublayer

The implementation of the interface between the Data Transfer Sublayer and the Channel Access Sublayer is not mandated or specified by this STANAG. Since the interface is internal to the subnetwork architecture and may be implemented in a number of ways it is considered beyond the scope of STANAG 5066. A model of the interface has been assumed, however, for the purposes of discussion and specification of other sublayer functions. The STANAG's presentation of the interface is modelled after that used in Reference 1 [1] to this STANAG, however, the interface defined in the reference is not mandated for use herein.

Despite the advisory nature of the conceptual model of the internal interface between the Data Transfer Sublayer and the Channel Access Sublayer, there are some mandatory requirements that are placed on any interface implementation.

The interface must support the service-definition for the Data Transfer Sublayer, i.e.:

1. The interface **shall** (1) allow the Channel Access Sublayer to submit protocol data units (i.e., C_PDUs) for transmission using the regular and expedited delivery services provided by the Data Transfer Sublayer.

2. The interface **shall** (2) allow the Data Transfer Sublayer to deliver C_PDUs to the Channel Access Sublayer.

3. The interface **shall** (3) permit the Channel Access Sublayer to specify the delivery services, priority, and time-to-die required by the C_PDUs when it submits them to the Data Transfer Sublayer.

4. The interface **shall** (4) permit the Data Transfer Sublayer to specify the delivery services that were used by received C_PDUs when it submits them to the Channel Access Sublayer.

5. The interface **shall** (5) permit the Channel Access Sublayer to specify the destination address to which C_PDUs are to be sent.

6. The interface **shall** (6) permit the Data Transfer Sublayer to specify the source address from which C_PDUs are received, and the destination address to which they had been sent.

7. The interface **shall** (7) permit the Data Transfer Sublayer to notify the Channel Access sublayer when a warning indication (i.e., a WARNING D_PDU) has been

[1] Clark, D., and N. Karavassillis, "Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio", TM-937, May 1998, pp. 109-115

received from a remote peer, the source and destination address associated with the warning, the reason for the warning, and the event (i.e., message type) that triggered the warning message.

8. The interface **shall** (8) permit the Data Transfer Sublayer to notify the Channel Access sublayer that a warning indication (i.e., a WARNING D_PDU) has been sent to a remote peer, the destination address associated with the warning, the reason for the warning, and the event (i.e., message type) that triggered the warning message.

9. The interface **shall** (9) permit the Channel Access sublayer to notify the Data Transfer Sublayer that a Connection (i.e., either an Exclusive or Nonexclusive Link) has been established with a given node.

10. The interface **shall** (10) permit the Channel Access sublayer to notify the Data Transfer Sublayer that a Connection (i.e., either an Exclusive or Nonexclusive Link) has been terminated with a given node.

11. The interface **shall** (11) permit the Data Transfer Sublayer to notify the Channel Access sublayer that a Connection (i.e., either an Exclusive or Nonexclusive Link) has been lost with a given node.

Additionally, the protocol-control information from the Channel Access sublayer that is required for the management of the Data Transfer Sublayer **shall** (12) not be derived from knowledge of the contents or format of any client data or U_PDUs encapsulated within the C_PDUs exchanged over the interface.
[Note: user's that encrypt their traffic prior to submittal may use the subnetwork. Subnetwork operation must be possible with client data in arbitrary formats that are unknown to the subnetwork, therefore any service requirements or indications must be provided by interface control information provided explicitly with the user data.]

The interface may use knowledge of the contents of C_PDUs (excluding the contents of any encapsulated U_PDUs) to derive protocol control information for the Data Transfer sublayer. This approach is highly discouraged, however. The recommended approach for implementation is that information required for protocol control within the Data Transfer sublayer should be provided explicitly in appropriate interface primitives.

In keeping with accepted practice in the definition of layered protocols, and as a means for specifying the operations of the sublayers that are mandated by this STANAG, the communication between the Data Transfer Sublayer and the Channel Access Sublayer is described herein with respect to a set of Primitives. The interface Primitives are a set of messages for communication and control of the interface and service requests made between the two layers.

By analogy to the design of the client-subnetwork interface specified in Annex A, the technical specification of the Data Transfer Sublayer assumes communication with the Channel Access Sublayer using primitives prefixed with a "D_". A minimal set of D_Primitives has been assumed that meet the requirements stated above and the general function for each D_Primitive is given in Table C-1. These D_Primitives are given without benefit of a list of arguments or detailed description of their use.

Table C-1 – Nominal Definition of D_Primitives for the Interface between the Data Transfer Sublayer and the Channel Access sublayer (non-mandatory, for information-only)

NAME OF PRIMITIVE	DIRECTION (see Note)	COMMENTS
D_UNIDATA_REQUEST	CAS →DTS	Request to send an encapsulated C_PDU using the regular delivery service to a specified destination node address with given priority, time-to-die, and delivery mode.
D_UNIDATA_REQUEST_CONFIRM	DTS →CAS	Confirmation that a given C_PDU has been sent using the regular delivery service
D_UNIDATA_REQUEST_REJECTED	DTS →CAS	Notification that a given C_PDU could not be sent using the regular delivery service and the reason why it was rejected by Data transfer Sublayer.
D_UNIDATA_INDICATION	DTS →CAS	Delivers a C_PDU that has been received using the regular delivery service from a remote node, with additional indications of the transmission service given to the C_PDU, the source node address, and the destination node address (e.g., if a group address is the destination).
D_EXPEDITED_UNIDATA_REQUEST	CAS →DTS	Request to send an encapsulated C_PDU using the expedited delivery service to a specified destination node address with a specified delivery mode.
D_EXPEDITED_UNIDATA_ REQUEST_CONFIRM	DTS →CAS	Confirmation that a given C_PDU has been sent using the expedited delivery service
D_ EXPEDITED_UNIDATA_ REQUEST_REJECTED	DTS →CAS	Notification that a given C_PDU could not be sent using the expedited delivery service and the reason why it was rejected by Data transfer Sublayer.
D_EXPEDITED_UNIDATA_ INDICATION	DTS →CAS	Delivers a C_PDU that has been received using the expedited delivery service from a remote node, with additional indications of the transmission service given to the C_PDU, the source node address, and the destination node address (e.g., if a group address is the destination).
D_WARNING_RECEIVED	DTS →CAS	Notifies the Channel Access sublayer that a WARNING D_PDU has been received from a remote node by the Data Transfer Sublayer, with the reason for sending the warning message
D_WARNING_TRANSMITTED	DTS →CAS	Notifies the Channel Access sublayer that a WARNING D_PDU has been sent to a remote node by the Data Transfer Sublayer, with the reason for sending the warning message
D_CONNECTION_MADE	CAS →DTS	Notifies the Data Transfer Sublayer that a connection has been made with a given remote node.
D_CONNECTION_TERMINATED	CAS →DTS	Notifies the Data Transfer Sublayer that a connection has been terminated with a given remote node.
D_CONNECTION_LOST	DTS →CAS	Notifies the Channel Access Sublayer that a connection has been lost with a given remote node.

Note: DTS = Data Transfer Sublayer; CAS = Channel Access Sublayer; <from sublayer> → <to sublayer>

C.3 Structure of Data Transfer Sublayer Protocol Data Units (D_PDUs)

In order to provide the data transfer services specified herein, the Data Transfer Sublayer

shall (1) exchange protocol data units (D_PDUs) with its peer(s).

The Data Transfer Sublayer **shall** (2) use the D_PDU types displayed in Table C-2 to support the **Selective ARQ service** and **Non ARQ** service, including the several data transfer submodes defined herein.

Table C-2. D_PDU Types

D_PDU Frame Types	*D_PDU Type #.*	*Function*	*Protocol Type*	*Frame Type*
DATA-ONLY	*0*	*Simplex data transfer*	*SRQ*	*I*
ACK-ONLY	*1*	*Acknowledgement of type 0, 2 data transfer*	-	*C*
DATA-ACK	*2*	*Duplex data transfer*	*SRQ*	*I + C*
RESET/WIN-RESYNC	*3*	*Reset/Re-synchronise peer protocol entities*	*IRQ*	*C*
EXP-DATA-ONLY	*4*	*Expedited simplex data transfer*	*SRQ*	*I*
EXP-ACK-ONLY	*5*	*Acknowledgement of type 4 data transfer*	-	*C*
MANANGEMENT	*6*	*Management message transfer*	*IRQ*	*C*
NON-ARQ DATA	*7*	*Non-ARQ data transfer*	*NRQ*	*I*
EXP NON-ARQ DATA	*8*	*Expedited non-ARQ data transfer*	*NRQ*	*I*
-	*9-14*	*Reserved for future extensions*	-	-
WARNING	*15*	*Unexpected or unrecognised D_PDU type*	-	*C*

C-Frame = Control Frame. I-Frame = Information Frame. I+C-Frame = Information + Control Frame.

All D_PDU types may be used to support half duplex and full duplex transmission modes, used by single and split-frequency operation. There are basically three different types of D_PDUs, or frames, noted by the *Frame-Type* field in Table C-2:

1. C (Control) Frames,
2. I (Information) Frames,
3. and a combined I+C Frame.

The *Protocol Type* field in Table C –2 indicates the type of data-transfer-service protocol with which the D_PDU frame **shall** (3) be used, as follows:

4. NRQ No Repeat-Request.(i.e., Non-ARQ) Protocol
5. SRQ Selective Repeat-Request Protocol
6. IRQ Idle Repeat-Request Protocol

The NRQ protocol **shall** (4) only operate in a simplex mode since the local node, after sending I-frames, does not wait for an indication from the remote node as to whether or not the I-frames were correctly received. Multiple repetitions of I-frames can be transmitted in order to increase the likelihood of reception under poor channel conditions, in accordance with the requested service characteristics.

The Selective RQ protocol **shall** (5) operate in a half or full duplex mode since the local node, after sending I-frames, waits for an indication in the form of a selective acknowledgement from the remote node as to whether the I-frames were correctly received or not. The local node then either sends the next I-frame, if all the previous I-frames were correctly received, or retransmits copies of the previous I-frame that were not, in accordance with the requirements of Section C.6. The local node will retransmit copies of the previous I-frames if no indication is received after a predetermined time interval. A local node sending I-frames in full duplex mode also sends the indication in the form of a selective acknowledgement embedded in the I-frames as to whether the I-frames were correctly received or not.

The Idle RQ protocol, also known as a stop and wait protocol, **shall** (6) operate in a half-duplex mode; the local node, after sending an I-frame, must wait until it receives an acknowledgement from the remote node as to whether or not the I-frame was correctly received. The local node then either sends the next I-frame, if the previous I-frame was correctly received, or retransmits a copy of the previous I-frame if it was not. The local node will retransmit a copy of the previous I-frame if no indication is received after a predetermined time interval.

Different D_PDU frame types may be combined in a transmission, subject to limitations imposed by the state of the Data Transfer Sublayer protocol. These states of the Data Transfer Sublayer protocol and their associated requirements are given in Section C.6.1

C.3.1 Generic simplified D_PDU structure

All D_PDU types that cannot carry segmented C_PDUs **shall** (1) be of the structure shown in Figure C-1 (a).

D_PDU types that can carry segmented C_PDUs **shall** (2) be structured according to Figure C-1 (b).

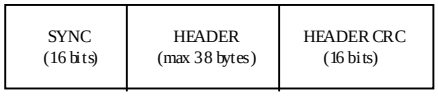


Figure C-1 (a). Format for D_PDU C-Frame types (1, 3, 5, 6 and 15)

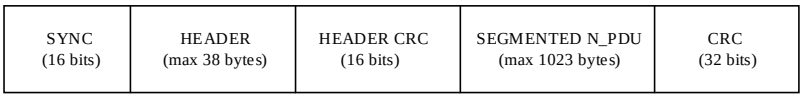


Figure C-1 (b). Format for D_PDU I and I+C Frame types (0, 2, 4, 7 and 8)

C.3.2 Generic detailed D_PDU structure

The detailed structure of the generic D_PDU C-Frame **shall** (1) be as shown in Figure C-2 (a) or Figure C-2(b)

The D_PDU types 1, 3, 5, 6 and 15 **shall** (2) use only the C-Frame structure defined in Figure C-2 (a).

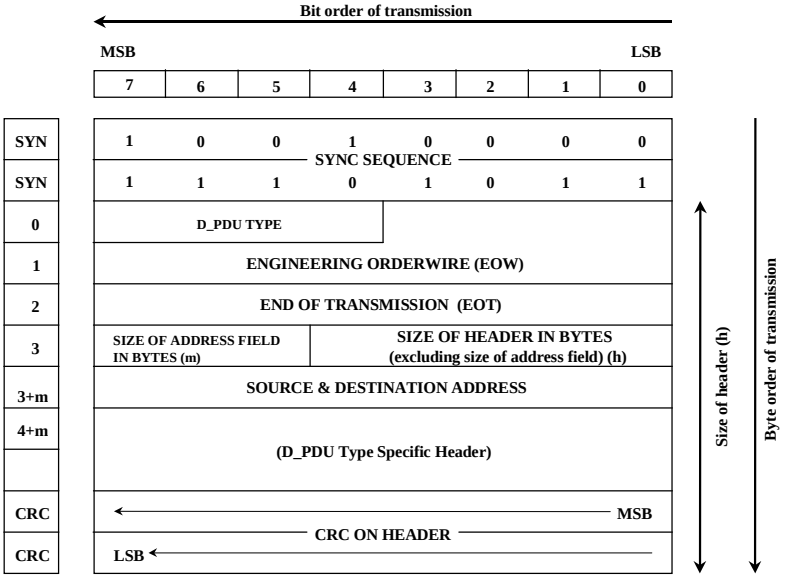


Figure C-2 (a). Generic D_PDU C-Frame Structure.

The D_PDU types 0, 2, 4, 7 and 8 **shall** (3) use the generic D_PDU I and I+C Frame structure defined in Figure C-2 (b).

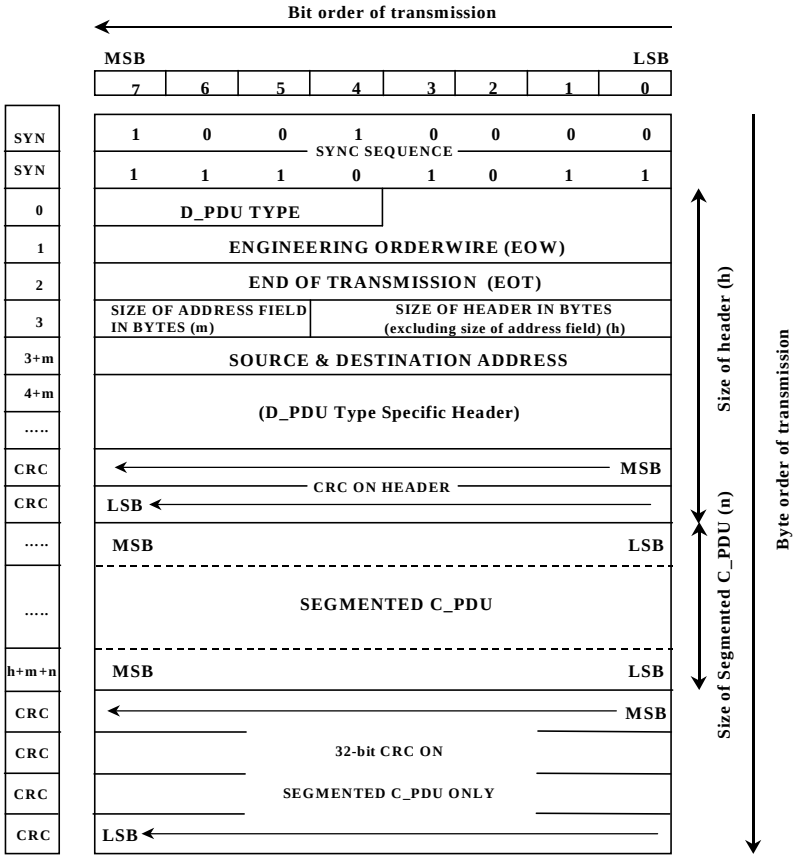


Figure C-2 (b). Generic D_PDU I and I+C Frame Structure.

All D_PDUs, regardless of type, **shall** (4) begin with the same 16-bit synchronisation (SYNC) sequence.

The 16-bit sequence **shall** (5) be the 16-bit Maury-Styles (0xEB90) sequence shown below, with the least significant bit (LSB) transmitted first:

(MSB) 1 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 (LSB)

16-bit Maury-Styles synchronisation sequence

The first 4 bytes of all D_PDU headers **shall** (6) contain the same fields:

1. a 4 bit *D_PDU Type* field that **shall** (7) identify the type of D_PDU;
2. a 12-bit field that **shall** (8) contain an engineering order wire (*EOW*) message;
3. an 8-bit field that **shall** (9) contain the end of transmission (*EOT*) information; and
4. a one byte field that **shall** (10) contain both a *Size-of-the-Address* field (3 bits) and a *Size-of-the-Header* (5 bits) field.

The next 1 to 7 bytes of every header, as specified in the *Size-of-the-Address* field, **shall** (11) contain source and destination address information for the D_PDU.

The D_PDU *Type-Specific-Header-Part* field **shall** (12) be as specified below in this STANAG, for each of the D_PDU types.

The last two bytes of every header **shall** (13) contain the Cyclic Redundancy Check (CRC) calculated in accordance with Section C.3.2.8.

The bits in any field in a D_PDU that is specified as NOT USED **shall** (14) contain the value zero (0).

C.3.2.1 D_PDU type

The D_PDU types **shall** (1) be as defined in Table C-2 and the D_PDU figures below.

The value of the D_PDU type number **shall** (2) be used to indicate the D_PDU type. The four bits available allow for 16 D_PDU types. Ten are here defined, the rest **shall** (3) remain reserved for future extensions to the STANAG.

C.3.2.2 Engineering Orderwire (EOW)

The 12 bit EOW field **shall** (1) carry Management messages for the Engineering Orderwire (EOW). EOW messages may not be explicitly acknowledged although the D_PDU of which they are a part may be. EOW messages can be explicitly acknowledged when they are contained in the MANAGEMENT Type 6 D_PDU through which Management-level acknowledgement services are provided in the Data Transfer Sublayer.

Figure C-3 (a) shows the generic 12-bit EOW structure. The first 4 bits of the EOW **shall** (2) contain the EOW-type field, which identifies the type of EOW message. The remaining 8-bits **shall** (3) contain the EOW-type-specific EOW data.

The various EOW messages including their definitions and extensions are specified further in Section C.5.

Figure C-3 (b) shows how the EOW message is mapped into the generic D_PDU header.

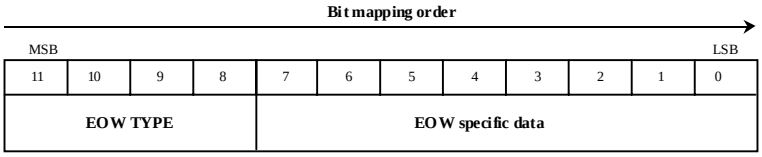


Figure C-3 (a). Generic EOW message.

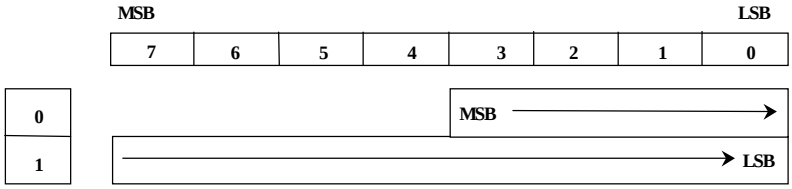


Figure C-3 (b). EOW mapping convention in D_PDU header.

C.3.2.3 End Of Transmission (EOT)

The 8-bit EOT field **shall** (1) provide an approximation of the time remaining in the current transmission interval specified by the transmitting node. This information is provided for the timing of exchanges between peer nodes to minimize collisions and the link turnaround time (time between the end of a transmission by one node and the start of a transmission by another node).

The number in this field **shall** (2) be a binary number expressing the number of half (1/2) second intervals remaining in the current transmission from the beginning of the current D_PDU including sync bytes.

When operating in half duplex mode, a node **shall** (3) not make a transmission during a transmission by another node (i.e., before the EOT expires).

Once an EOT is sent, the EOT in each subsequent D_PDU in that transmission **shall** (4) contain a consistent calculation of the EOT, that is, monotonically decreasing in half-second (0.5 second) intervals.

Calculations of the EOT by a transmitting node **shall** (5) be rounded up to the nearest half-second interval. [Note: rounding-up the calculation of the EOT, rather than truncating it, ensures that the transmission will be completed by the time declared in the EOT field of the D_PDU.]

A half-duplex node **shall** (6) stop transmitting and shift to receive mode when the EOT becomes zero.

When a node is in broadcast mode, the EOT field **shall** (7) be filled with all zeros, unless the remaining broadcast transmission interval is within the maximum EOT value of 127.5 seconds. If a node is in broadcast mode and either the remaining broadcast transmission

interval or the total broadcast transmission interval is within the maximum EOT value of 127.5 seconds, the EOT value **shall** (8) be computed and advertised as specified herein.

When a node is configured for and operating in full duplex mode, the EOT field **shall** (9) be filled with all zeros.

When D_PDU types are combined in a single transmission by the sublayer, any previously advertised values of EOT **shall** (10) not be violated or contradicted.

When D_PDU types are combined in a single transmission by the sublayer, advertised values for EOT **shall** (11) refer to the end of the combined transmission and not to the end of transmission of the D_PDU which contains the EOT field.

C.3.2.4 Size of Address Field

The Size-of-Address Field **shall** (1) specify the number of bytes in which the source and destination address are encoded (Note: this value is denoted by the integer value "m" in Figure C-2(a) and Figure C-2(b)). The address field may be from one (1) to seven (7) bytes in length, with the source and destination address of equal length.

Since the D_PDU header must be made up of an integer number of bytes, addresses **shall** (2) be available in 4-bit increments of size: 4 bits (or 0.5 bytes), 1 byte, 1.5 bytes, 2 bytes, 2.5 bytes, 3 bytes, and 3.5 bytes.

C.3.2.5 Size of Header Field

The Size-of-Header field **shall** (1) specify the number of bytes in which the D_PDU is encoded. (Note: this value is denoted by the integer value "h", Figure C-2(a) and Figure C-2(b)), and its value includes the sizes of the following fields and elements:

- D_PDU Type
- EOW
- EOT
- Size of Address field
- Size of Header field
- D_PDU-Type-specific header
- CRC field

The value of the Size-of-Header field **shall** (2) not include the size of the source and destination address field.

C.3.2.6 Source & Destination Address

Each D_PDU transmitted by a node **shall** (1) contain the source and destination address. Half of the bits are assigned to the source and the other half to the destination.

The first half **shall** (2) be the destination address and the second half **shall** (3) be the source address as displayed nominally in Figure C-4(a) (which assumes an odd-number as the address-field size) or Figure C-4(b) (which assumes an even-number as the address-field size).

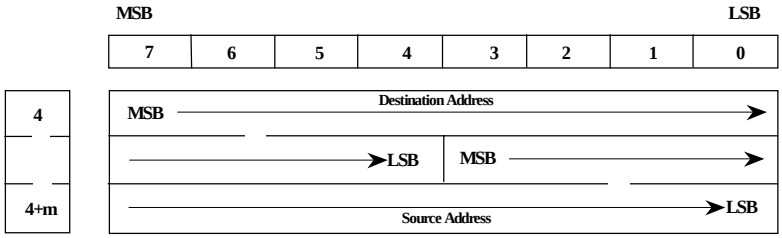


Figure C-4(a). Address mapping convention in D_PDU header, assuming address-field size is odd.

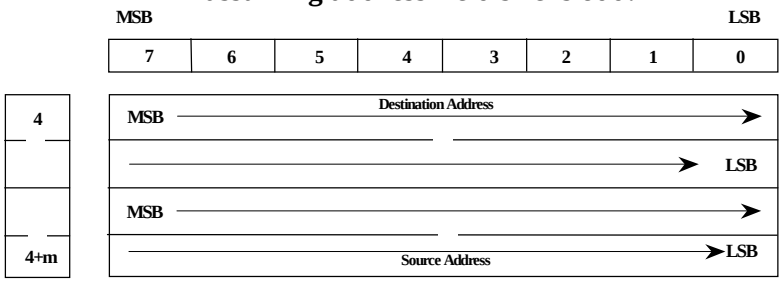


Figure C-4(a). Address mapping convention in D_PDU header, assuming address-field size is even.

Addresses **shall** (4) be in the form of a binary number. With 7 bytes available for each of the user and the destination, the smallest possible address field is 4 bits, the largest possible is 3.5 bytes, or 28 bits. This allows more than 268 million addresses, if the maximum field size is used.

A decimal number **shall** (5) represent each byte or fractional byte of an address, and the binary equivalent **shall** (6) be mapped into the corresponding byte.

Any fractional-byte elements in the address **shall** (7) be mapped into the first (leftmost) non-zero number in the in the decimal representation of the address. The remaining numbers in the decimal representation of the address **shall** (8) refer to byte-sized elements in the address field.

[Note: As an example, if 3.5 bytes are used, the address would be expressed as w.x.y.z, where w can be any value from 0 to 15, and x, y and z can be any value from 0 to 255. The value w will represent the most significant bits and z the least significant bits of the address.]

The address bits **shall** (9) be mapped into the address field by placing the MSB of the address into the MSB of the first byte of the address field, and the LSB into the LSB of the last byte of the field, in accordance with Figure C-4(a), for addresses with length of 0.5, 1.5, 2.5, or 3.5 bytes, and Figure C-4(b), for addresses with length of 1, 2, or 3 bytes.

When a field spans more than one octet, the order of the bit values within each octet **shall** (10) decrease progressively as the octet number increases.

The lowest bit number associated with the field represents the lowest-order value. Leading address bytes which are zero may be dropped from the address, consistent that the requirement that the source and destination address subfields must be of equal length. Trailing address bytes that are zero **shall** (11) be sent.

C.3.2.7 D_PDU Type-Specific Header

The bytes immediately following the address field **shall** (1) encode the D_PDU Type-Specific header, as specified in the corresponding section below from Sections C.3.3 through C.3.12.

C.3.2.8 Cyclic Redundancy Check (CRC)

The two-bytes following the D_PDU Type-Specific header **shall** (1) contain a 16-bit Cyclic Redundancy Check (CRC) field.

The header CRC error-check field **shall** (2) be calculated using the following polynomial: $x^{16} + x^{15} + x^{12} + x^{11} + x^8 + x^6 + x^3 + 1$, or in hexadecimal format 0x19949, using the shift-register method shown by the figures in Appendix I of CCITT Recommendation V.41 (or equivalent method in software; an example is given below).

[Note: This polynomial is not the same as that shown in the figure in V.41, Appendix I; the polynomial was chosen to provide a lower probability of undetected error (i.e., better performance) than that given by the polynomial specified in V.41. The polynomial specified here was analyzed and reported by Wolf in a paper in the 1988 IEEE Conference on Military Communications (MILCOM- paper 15.2). Note too that the taps in the figure are shown reversed from the order in which they occur in the polynomial defined in V.41. This reversal is accomodated by the remaining requirements specified below.]

When calculating the header CRC field, the shift registers **shall** (3) be initially set to all (0) zeros.

The header CRC **shall** (4) be calculated over all bits in the header, excluding the Maury-Styles synchronisation sequence, and including the following fields and elements:

- D_PDU Type
- EOW
- EOT
- Size of Address field
- Size of Header field
- Source and Destination Address
- D_PDU-Type-Specific header

A node **shall** (5) process the information contained in a header with a valid CRC, regardless of the result of the CRC error check over any segmented C_PDU that may be a part of the D_PDU.

The CRC bits **shall** (6) be mapped (see Figure C-5) into the CRC octets by placing the MSB of the CRC into the LSB of the first byte of the CRC field, and the LSB of the CRC into the MSB of the last byte of the CRC field. This will result in the MSB of the most significant byte of the CRC being sent first, followed by the remaining bits in descending order, which is consistent with the order of transmission for CRC bits, as specified in Recommendation V.42, section 8.1.2.3.

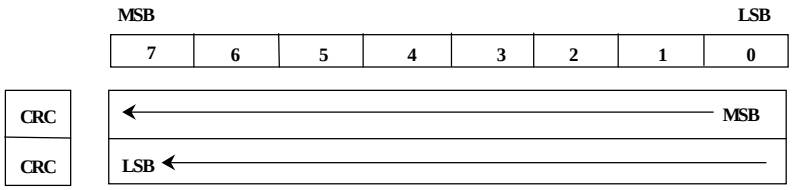


Figure C-5. CRC mapping convention in D_PDU header.

The following C code can be used to calculate the CRC value using the specified polynomial.

```
        /* Polynomial x^16 + x^15 + x^12 + x^11 + x^8 + x^6 + x^3 + 1        */
unsigned short CRC_16_S5066(unsigned char DATA, unsigned short CRC)
{
 unsigned char i, bit;
 for (i=0x01; i; i<<=1)
 {
 bit = (((CRC & 0x0001) ? 1:0)^((DATA&i) ? 1:0));
 CRC>>=1;
 if (bit) CRC^=0x9299;                  /* polynomial representation, bit */
 }                                      /* -reversed (read right-to-left), and */
                                        /* with the initial x^16 term implied.  */
 return (CRC);
}

/*   example of usage: */
#define NUM_OCTETS;       /* number of octets in the message data */
unsigned char        message[NUM_OCTETS];
unsigned short       CRC_result;
unsigned int  j;

CRC_result = 0x0000;
for (j=0; j < NUM_OCTETS; j++){
        CRC_result = CRC_16_S5066(message[j], CRC_result);
        }
/* CRC_result contains final CRC value when loop is completed */
```

NOTE: *This code calculates the CRC bytes in the proper order for transmission as defined above; no bit reversal is required with this code.*

Code Example C-1. Calculation of the CRC-16-S5066 for D_PDU Headers.

The function CRC_16_5066 in Code Example C-1 may be used to calculate the CRC for a data sequence of octets, and is called for each successive octet in the sequence. The function accepts as its first argument an octet in the data sequence, and as its second argument, the value of the CRC calculated for the previous octet in the sequence. The function returns the value of the CRC. When called for the first octet in the data sequence, the value of the CRC must be initialized to zero as required.

The result of this code for the D_PDU described below is 0x1E5F. The least significant byte of the computed CRC should be transmitted before the most significant byte; as noted, the algorithm already performs the requisite bit-level reversal. The receive processing should extract the CRC bytes and re-assemble into the correct order.

D_PDU used in above CRC calculation:

D_PDU Type:	Warning
EOW:	Type 0 (i.e., all zeros)
EOT:	all zeros
Address size:	2
Destination address:	0.0.0.5 (leading zeros not encoded)
Source address:	0.0.0.100 (leading zeros not encoded)
Received D_PDU type:	0
Reason warning sent:	2
CRC:	0x1E5F

The full D_PDU is then (including sync bytes and CRC-field, properly bit-reversed and in the order they should be transmitted):

0x90, 0xEB, 0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02, 0x5F, 0x1E

C.3.2.9 Segmented C_PDU

For the I and I+C D_PDUs types, the octets of the segmented C_PDUs **shall** (1) be transmitted in ascending numerical order, following the two-byte CRC on the D_PDU header.

Within an octet, the LSB **shall** (2) be the first bit to be transmitted as shown in Figure C-6.

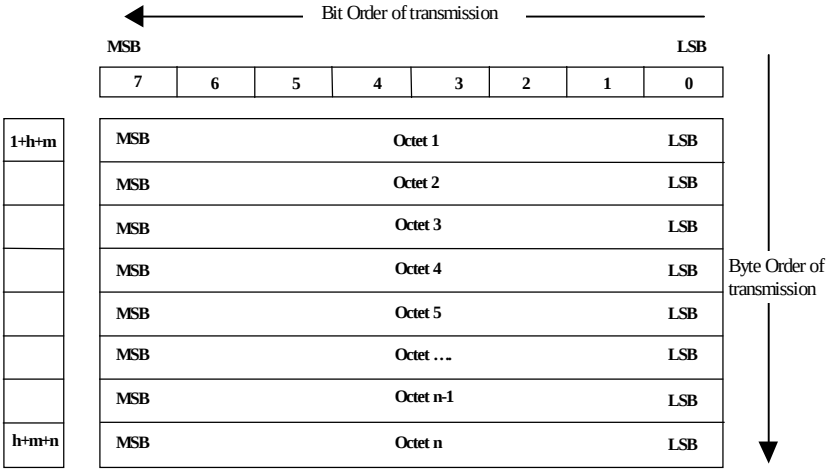


Figure C-6. Segmented C_PDU mapping convention in D_PDU structure.

C.3.2.10 Size of Segmented C_PDU

The SIZE OF SEGMENTED C_PDU field **shall** (1) be used only with D_PDUs that are I or I+C frame types, i.e, that have a Segmented C_PDU field as shown in Figure C-2(b). (Note: The value of the SIZE OF SEGMENTED C_PDU field is denoted by the integer value "n" in the Figure). This field actually is contained within the D_PDU Type-Specific Header part, but since it is common to all I and I+C D_PDUs the format is defined here.

The bit-value of the SIZE OF SEGMENTED C_PDU **shall** (2) be encoded as a ten-bit field as indicated by Figure C-7.

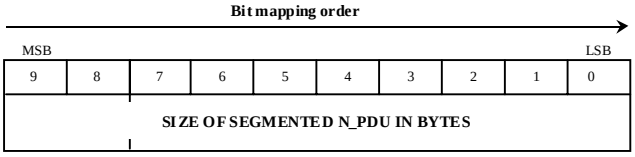


Figure C-7. SIZE OF SEGMENTED C_PDU Field.

The value in the SIZE OF SEGMENTED C_PDU field **shall** (3) not include the two-bytes for the CRC following the Segmented C_PDU. The Segmented C_PDU field can hold a maximum of 1023 bytes from the segmented C_PDU.

The SIZE OF SEGMENTED C_PDU **shall** (4) be mapped into consecutive bytes of the D_PDU as indicated in Figure C-8, in the byte locations specified for the applicable D_PDU.

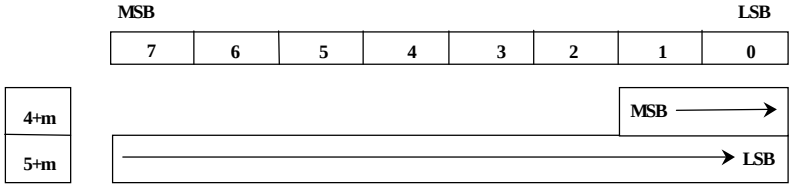


Figure C-8. Segmented C_PDU Size mapping convention in all applicable D_PDU header structures.

C.3.2.11 CRC-ON-SEGMENTED-C_PDU Field

The last four bytes of any I or I+C D_PDU **shall** (1) contain a 32-bit Cyclic Redundancy Check (CRC) field.

The CRC **shall** (2) be applied and computed on the contents of the Segmented C_PDU using the following polynomial [see footnote [2] below]:

$$x^{32} + x^{27} + x^{25} + x^{23} + x^{21} + x^{18} + x^{17} + x^{16} + x^{13} + x^{10} + x^{8} + x^{7} + x^{6} + x^{3} + x^{2} + x + 1$$

or, in hexadecimal notation: 0x10AA725CF,

using the shift-register method similar to that shown by the figures in Appendix I of CCITT Recommendation V.41, but using a longer shift register and appropriate changes to the tap configuration corresponding to the polynomial specified. Equivalent implementations in software may be used to compute the 32-bit CRC (an example is offered below).

When calculating the header CRC field, the shift registers **shall** (4) be initially set to all (0) zeros.

[2] This polynomial is constructed with a methodology similar to that used to construct the 16-bit polynomial on the header, and for the same reason. Using this methodology, the polynomial for the 32-bit CRC on the segmented C_PDU is the generator polynomial for a two-error-correcting BCH code (65535,32), with a maximum information block length of 65503 bits. Only the error-detection properties of the code are used.

The following C code can be used to calculate the CRC value using the specified polynomial.

```
/* Polynomial:
   x^32 + x^27 + x^25 + x^23 + x^21 + x^18 + x^17 + x^16 + x^13 + x^10 + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1 */
    unsigned int CRC_32_S5066(unsigned char DATA, unsigned int CRC)
    {
     unsigned char i, bit;

     for (i=0x01; i; i<<=1)
     {
     bit = (((CRC & 0x0001) ? 1:0)^((DATA&i) ? 1:0));
     CRC>>=1;
     if (bit) CRC^=0xF3A4E550;              /* polynomial representation, bit */
     }                                      /* -reversed (read right-to-left), and */
                                            /* with the initial x^32 term implied.  */
     return (CRC);
    }

    /*   example of usage: */
    #define NUM_OCTETS;          /* number of octets in the message data */
    unsigned char  message[NUM_OCTETS];
    unsigned int    CRC_result;
    unsigned int    j;

    CRC_result = 0x00000000;
    for (j=0; j < NUM_OCTETS; j++){
            CRC_result = CRC_32_S5066(message[j], CRC_result);
            }
    /* CRC_result contains final CRC value when loop is completed */
```

NOTE: *This code calculates the CRC bytes in the proper order for transmission as defined above; no bit reversal is required with this code.*

Code Example C-2. Calculation of the CRC-32-S5066 on Segmented C_PDUs.

The function CRC_32_S5066 in Code Example C-2 may be used to calculate the CRC for a data sequence of octets, and is called for each successive octet in the sequence. It is initialized and used in the same manner as the example for the STANAG 5066 16-bit CRC.

When applied to this short C_PDU message data sequence:

message[] = {0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02},

the result is the 32-bit value (in hexadecimal format): 0xF4178F95

Just as for the 16-bit CRC computation, the algorithm that calculates the 32-bit CRC performs the bit-level reversal in place, and the resultant value must be transmitted least-significant byte first, in order to most-significant byte last. The full message data, with 32-bit CRC appended in proper position, is the following sequence:

0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02, 0x95, 0x8F, 0x17, 0xF4

C.3.3 DATA-ONLY (TYPE 0) D_PDU (Simplex data transfer)

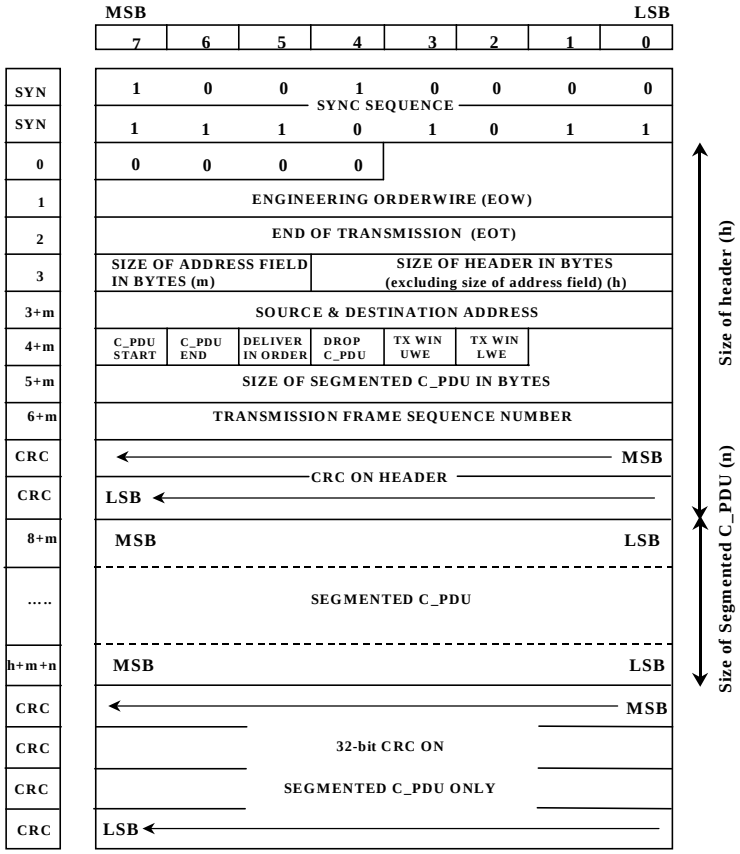


Figure C-9. Frame format for DATA-ONLY D_PDU Type 0

The DATA-ONLY D_PDU **shall** (1) be used to send segmented C_PDUs when the transmitting node needs an explicit confirmation the data was received.

The DATA-ONLY D_PDU **shall** (2) be used in conjunction with a basic selective automatic repeat request type of protocol.

A Data Transfer Sublayer entity that receives a DATA-ONLY D_PDU **shall** (3) transmit an ACK-ONLY (TYPE 1) D_PDU or a DATA-ACK (TYPE 2) D_PDU as acknowledgement, where the type of D_PDU sent depends on whether or not it has C_PDUs of its own to send to the source of the DATA-ONLY D_PDU.

The DATA-ONLY D_PDU **shall** (4) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-9 and the paragraphs below:

- C_PDU START
- C_PDU END
- DELIVER IN ORDER
- DROP C_PDU
- TX WIN UWE
- TX WIN LWE
- SIZE OF SEGMENTED C_PDU
- TRANSMIT SEQUENCE NUMBER

The C_PDU START flag **shall** (5) be set to indicate the start of a newly segmented C_PDU; the C_PDU segment contained within this D_PDU is the first segment of the C_PDU, in accordance with the C_PDU-segmentation process described in section C.4

The C_PDU END flag **shall** (6) be set to indicate the end of a segmented C_PDU; when a D_PDU is received with the C_PDU END flag set it indicates the last D_PDU that was segmented from the C_PDU. Depending on the status of the DELIVER IN ORDER flag, the link layer will assemble and deliver the C_PDU, if all D_PDUs between and including the C_PDU START and C_PDU END are received completely error free. The re-assembly process of D_PDUs into C_PDUs is described in section C.4

If the DELIVER IN ORDER flag is set on the D_PDUs composing a C_PDU, the C_PDU **shall** (7) be delivered to the upper layer when both the following conditions are met:

1) The C_PDU is complete.
2) All C_PDUs received previously that also had the DELIVER IN ORDER flag set have been delivered.

If the DELIVER IN ORDER flag is cleared on the D_PDUs composing a C_PDU, the C_PDU **shall** (8) be delivered to the upper layer when the following condition is met:

3) The C_PDU is complete and error free.

When the N_DROP PDU flag is set by the D_PDU source, the receiving Data Transfer Sublayer **shall** (9) discard the contents of the segmented C_PDU field of the current D_PDU and all other previously received segments of the C_PDU of which the current D_PDU is a part.

No segmented C_PDU data needs to be sent if the N_DROP PDU flag is set and the SIZE OF SEGMENTED C_PDU field **shall** (10) be zero in this case.

The TX WIN UWE flag **shall** (11) be set when the TRANSMIT FRAME SEQUENCE NUMBER for the current D_PDU is equal to the Transmit Window Upper Edge (TX UWE) of the transmit-flow-control window.

Similarly, the TX WIN LWE flag **shall** (12) be set when the TRANSMIT FRAME SEQUENCE NUMBER for the current D_PDU is equal to the Transmit Lower Window Edge (LWE) of the transmit flow control window.

The SIZE OF SEGMENTED C_PDU field **shall** (13) be encoded as specified in Section C.3.2.10.

The TRANSMIT FRAME SEQUENCE NUMBER field **shall** (14) contain the sequence number of the current D_PDU.

The value of the TRANSMIT FRAME SEQUENCE NUMBER field **shall** (15) be a unique integer (modulo 256) assigned to the D_PDU during the segmentation of the C_PDU, and will not be released for reuse with another D_PDU until the receiving node has acknowledged the D_PDU.

Values for the TRANSMIT FRAME SEQUENCE NUMBER field **shall** (16) be assigned in an ascending modulo 256 order during the segmentation of the C_PDU.

The SEGMENTED C_PDU field **shall** (17) immediately follow the D_PDU header as depicted in Figure C-7. Segmented C_PDUs **shall** (18) be mapped according to the specification of Section C.3.2.9.

C.3.4 ACK_ONLY (TYPE 1) D_PDU (Acknowledgement of type 0, 2 data transfer)

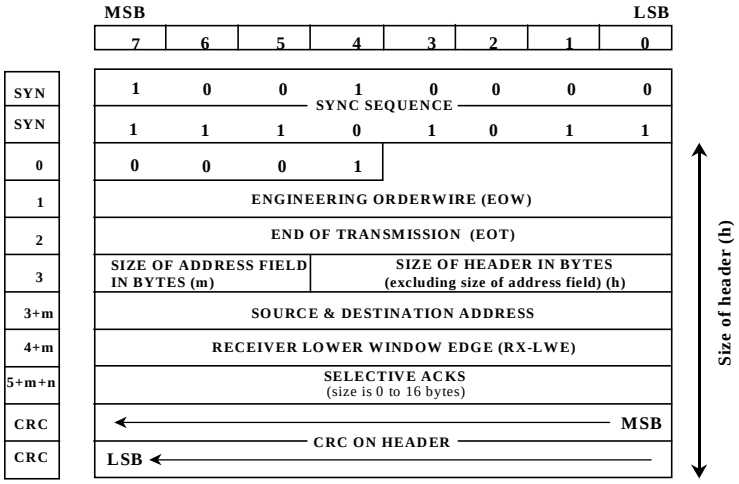


Figure C-10. Frame format for ACK-ONLY D_PDU Type 1

The ACK-ONLY D_PDU **shall** (1) be used to selectively acknowledge received DATA-ONLY or DATA-ACK D_PDUs when the receiving Data Transfer Sublayer has no segmented C_PDUs of its own to send.

The ACK-ONLY D_PDU **shall** (2) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-10 and the paragraphs below:

- RECEIVE LOWER WINDOW EDGE (LWE)
- SELECTIVE ACKS

The value of the RECEIVE LOWER WINDOW EDGE (RX LWE) field **shall** (3) equal the D_PDU sequence number (modulo 256) of the RX LWE pointer associated with the node's receive ARQ flow-control window**.**

The SELECTIVE ACKS field can have a dynamic length of 0 to 16 bytes, and **shall** (4) contain a bit-mapped representation of the status of all received D_PDUs with sequence numbers from the LWE to and including the UWE pointers of the receive flow-control window. The size of the SELECTIVE ACK field can be determined from knowledge of the structure of the ACK-ONLY D_PDU and the value of the Size of Header field, and is not provided explicitly in the ACK-ONLY D_PDU.

A set (1) bit within the SELECTIVE ACKS field **shall** (5) indicate a positive acknowledgement (ACK), i.e., that the D_PDU with the corresponding Frame Sequence Number was received correctly.

Only D_PDU frames with a correct segmented C_PDU CRC **shall** (6) be acknowledged positively even if the header CRC is correct.

Frames with the DROP C_PDU flag set **shall** (7) be acknowledged positively regardless of the results of the CRC check on the segmented C_PDU.

A cleared (0) bit within the SELECTIVE ACKS field **shall** [8] indicate a negative acknowledgement (NACK), i.e., that the D_PDU with the corresponding Frame Sequence Number was received incorrectly, or not at all.

The construction of the SELECTIVE ACK field and the mapping of D_PDU frame-sequence numbers to bits within the SELECTIVE ACK field **shall** [9] be in accordance with Figure C-11 and Figure C-12 and the paragraphs below.

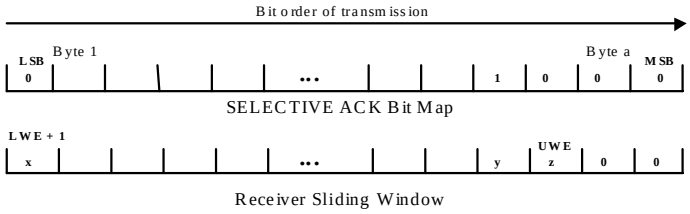


Figure C-11. Constructing the SELECTIVE ACK Field

(Note: the bits that are set (1) and cleared (0) are representing example bits.)

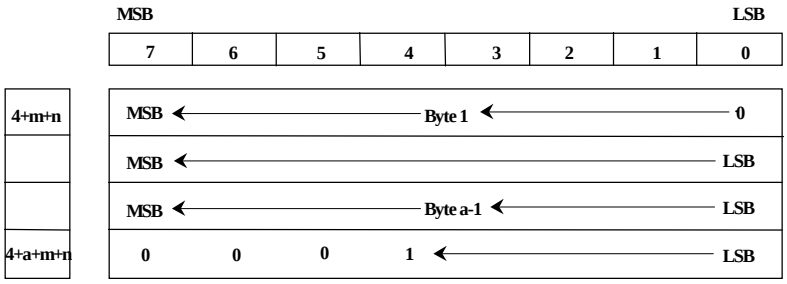


Figure C-12. SELECTIVE ACK mapping convention

(Note: the example bits that were shown in figure L-5 can been seen in this mapping example)

The LSB of the first byte of the SELECTIVE ACK field **shall** [10] correspond to the D_PDU whose frame sequence number is equal to 1 + the value in the RX LWE field of this ACK-ONLY D_PDU [the bit corresponding to the RX LWE is always zero, by definition, and is therefore not sent].

Each subsequent bit in the SELECTIVE ACK field **shall** [11] represent the frame sequence number of a subsequent D_PDU in the receive flow-control sliding window, in ascending order of the respective frame-sequence numbers without omission of any values.

The bit corresponding to the Receive Upper Window Edge (RX UWE) **shall** [12] be in the last byte of the SELECTIVE ACK field.

If the bit representing the RX UWE is not the MSB of the last byte, the remaining bits in the byte (until and including the MSB) **shall** [13] be set to 0 as padding. No further SELECTIVE ACK bytes **shall** [14] be transmitted after such bits are required.

C.3.5 DATA-ACK (TYPE 2) D_PDU (Duplex data transfer)

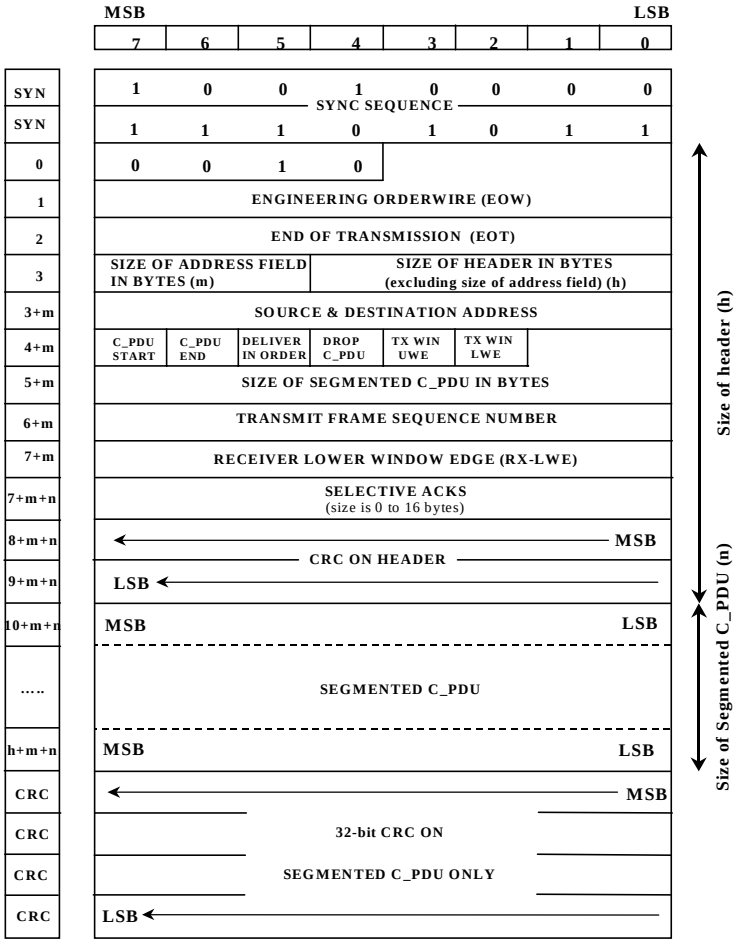


Figure C-13. Frame format for DATA-ACK D_PDU Type 2

The DATA-ACK (TYPE 2) D_PDU is a part of a basic selective automatic repeat request type of protocol. The DATA-ACK D_PDU **shall** (1) be used to send segmented C_PDUs when the transmitting node needs an explicit confirmation the data was received and has received D_PDUs to selectively acknowledge.

A Data Transfer Sublayer entity that receives a DATA-ACK D_PDU **shall** (2) transmit an ACK-ONLY (TYPE 1) D_PDU or a DATA-ACK (TYPE 2) D_PDU as acknowledgement, where the type of D_PDU sent depends on whether or not it has C_PDUs of its own to send to the source of the DATA-ACK D_PDU.

The DATA-ACK D_PDU is a combination of the DATA-ONLY and ACK-ONLY D_PDU. All of the field specifications from the DATA-ONLY and ACK-ONLY D_PDUs apply to this D_PDU. The DATA-ACK D_PDU **shall** (4) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-13 and the referenced paragraphs:

- C_PDU START – **shall** (5) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- C_PDU END– **shall** (6) be as specified in Section 3.3 for the DATA-ONLY D_DPU;

- DELIVER IN ORDER– **shall** (7) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- DROP C_PDU– **shall** (8) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- TX WIN UWE– **shall** (9) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- TX WIN LWE– **shall** (10) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- SIZE OF SEGMENTED C_PDU– **shall** (11) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- TRANSMIT SEQUENCE NUMBER– **shall** (12) be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- RECEIVE LOWER WINDOW EDGE (RX LWE) – **shall** (13) be as specified in Section 3.4 for the ACK-ONLY D_DPU;
- SELECTIVE ACKS– **shall** (14) be as specified in Section 3.4 for the ACK-ONLY D_DPU;

C.3.6 RESET/WIN_RESYNC (TYPE 3) D_PDU (Reset/Re-synchronise peer protocol entities)

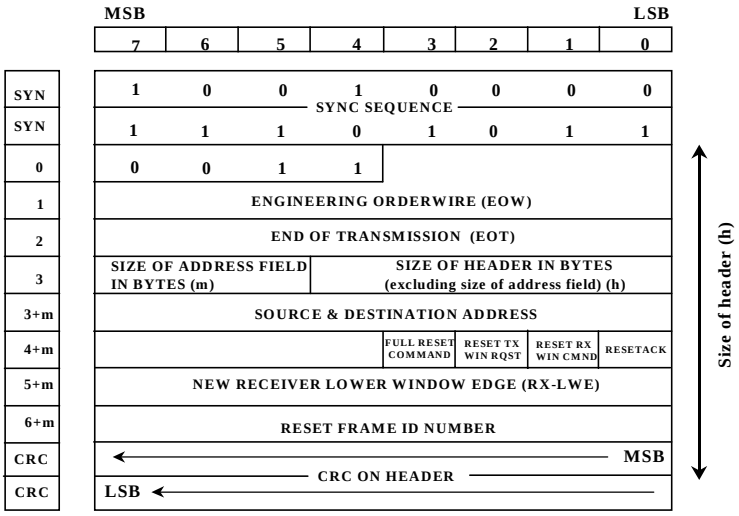


Figure C-14. Frame format for RESET/WIN RESYNC D_PDU Type 3

The RESET/WIN RESYNC D_PDU **shall** (1) be used to control the re-synchronisation or re-initialisation of the selective-repeat ARQ protocol operating on the link between the source and destination nodes.

Reset and resynchronization operations **shall** (2) be performed with respect to the transmit lower-window edge (TX LWE) and the receive lower-window edge (RX LWE) for the flow-control sliding windows at the sending node and receiving node, as specified in Section C.6.2.

The RESET/WIN RESYNC D_PDU **shall** (3) use a basic stop and wait type of protocol (denoted as the IRQ protocol elsewhere in this STANAG) in which the reception of this D_PDU **shall** (4) result in the transmission of an acknowledgement D_PDU by the receiving node. For the IRQ protocol used with the RESET/WIN RESYNC D_PDU, the RESET/WIN RESYNC D_PDU is used for both data and acknowledgement, as specified below.

Transmission of D_PDUs supporting the regular-data service, i.e., of DATA (Type 0), ACK (Type 1), and DATA-ACK (Type 2) D_PDUs, **shall** (5) be suspended pending completion of any stop-and-wait protocol using the RESET/WIN RESYNC D_PDUs. [Note: The ARQ windowing variables will be reset to zero or changed as a result of the use of the REST/WIN RESYNC D_PDUs, and suspension of normal data transmission until these variables have been reset is required, lest the variables controlling the selective-repeat ARQ protocol become even further corrupted than the error-state that presumably triggered the reset or resynchronization protocol.]

The RESET/WIN RESYNC D_PDU **shall** (6) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-14 and the paragraphs below:

- FULL RESET COMMAND
- RESET TX WIN RQST
- RESET TX WIN CMND
- RESET ACK
- NEW RECEIVE LOWER WINDOW EDGE (LWE)
- RESET FRAME ID NUMBER

The FULL RESET COMMAND flag **shall** (7) be set equal to one (1) to force a full reset of the ARQ machines at the transmitter and receiver to their initial values as specified in Sections C.6.2 and C.6.3.

A Type 3 D_PDU with the RESET TX WIN RQST flag set equal to one (1) **shall** (8) be used to request a resynchronisation of the TX-LWE and RX-LWE pointers used for DATA in the transmit and receive nodes.

A node that receives a Type 3 D_PDU with the RESET TX WIN RQST flag set equal to one **shall** (9) respond by forcing resynchronization of the windows using a RESET/WIN RESYNC D_DPU and the RESET TX WIN CMND flag, as specified below.

A RESET/WIN RESYNC TYPE 3 D_PDU with the RESET TX WIN CMND flag set equal to one (1) **shall** (10) be used to force a resynchronisation of the TX-LWE and RX-LWE pointers.

A node that sends a Type 3 D_PDU with the RESET TX WIN CMND flag set equal to one **shall** (11) proceed as follows:

- The NEW RECEIVE LWE field **shall** (12) be set equal to the value of the sending node's RX-LWE.
- The sending node **shall** (13) wait for a RESET/WIN RESYNC Type 3 D_PDU with the RESET ACK flag set equal to one as an acknowledgement that the resynchronization has been performed.

A node that receives a Type 3 D_PDU with the RESET TX WIN CMND flag set equal to one **shall** (14) proceed as follows:

- The value of the node's TX LWE **shall** (15) be set equal to the value of the NEW RECEIVE LWE field in the RESET/WIN RESYNC D_PDU that was received;
- The node **shall** (16) send a RESET/WIN RESYNC Type 3 D_PDU with the RESET ACK flag set equal to one as an acknowledgement that the resynchronization has been performed.

RESETACK flag **shall** (17) be set equal to one (1) to indicate an acknowledgement of the most recently received RESET/WIN RESYNC TYPE 3 D_PDU.

A node may use a RESET/WIN RESYNC acknowledgement transmission, i.e., a Type 3 D_PDU with the RESETACK flag set equal to one (1), to request/force a reset or resynchronization of its own ARQ flow control variables, e.g., if the link is supporting reliable duplex communication and the ARQ machines for both directions of data-flow must be reset or resynchronized.

The NEW RECEIVE LWE field specifies the value of the new receiver ARQ RX-LWE, as

noted above, and **shall** (18) be valid only when the value of the RESET WIN CMND flag equals one (1). The value of the NEW RECEIVE LWE field **shall** (19) be ignored in any other situation.

The Data Transfer Sublayer **shall** (20) use the RESET FRAME ID NUMBER field to determine if a given RESET/WIN RESYNC D_PDU received is a copy of one already received.

The value of the RESET FRAME ID NUMBER field **shall** (21) be a unique integer (modulo 256) assigned in ascending order to RESET/WIN RESYNC D_PDUs, and will not be released for reuse with another D_PDU until the D_PDU to which it was assigned has been acknowledged.

[Implementation Note: Due to the fact that C_PDUs have a certain time-to-live (TTL) assigned to them it is possible that this time passes when the C_PDU is partially transmitted and acknowledged by the transmitting node. Deleting such partially transferred C_DPU can be performed by two methods:

a) The first method makes use of a D_DPU frame's DROP C_PDU flag, setting this flag will indicate to the receiving node that the current D_PDU and all other previously received portions of the C_PDU have to be discarded by the link layer. This method still requires the transmission and acknowledgement of all D_PDU headers only until the dropped C_PDU has been completely received and acknowledged.

b) The second method makes use of the RESET/WIN RESYNC D_PDU, sending this D_PDU will enable the transmitter to skip the discarded C_PDU or multiple C_PDUs without the need of sending all D_PDU headers. It is also possible to RESET the Selective ARQ machine by sending a FULL RESET COMMAND. This will put the ARQ machine back into its starting position upon the creation of a physical link. The FULL RESET COMMAND will not delete one or more selected C_PDUs but will delete all partially received and acknowledged C_PDUs still waiting to be completed.]

C.3.7 EXPEDITED DATA ONLY (TYPE 4) D_PDU (Simplex expedited data transfer)

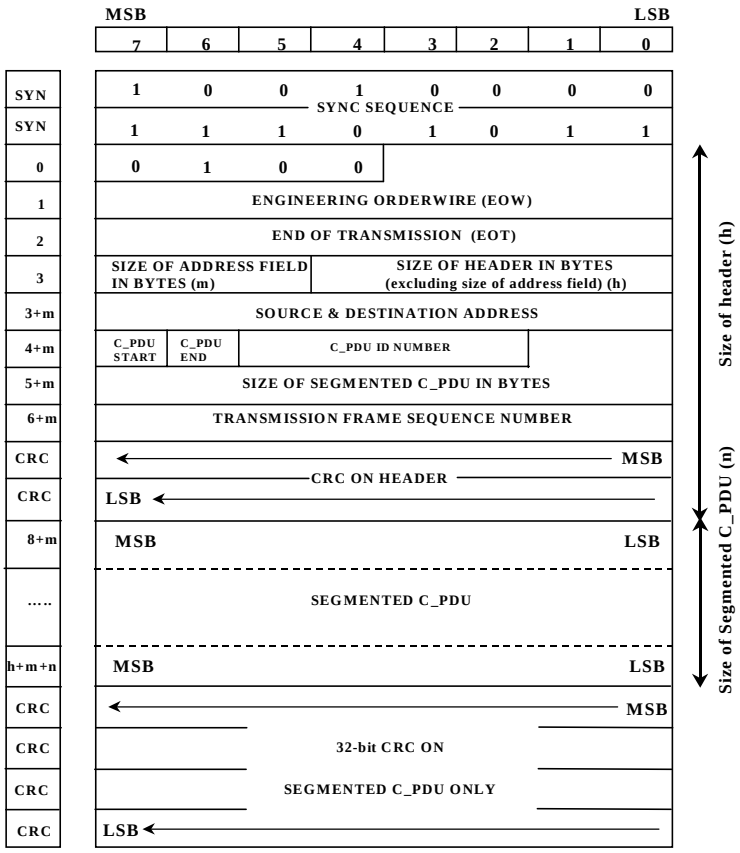


Figure C-15. Frame format for EXPEDITED DATA ONLY D_PDU Type 4

The EXPEDITED DATA-ONLY (TYPE 4) D_PDU **shall** (1) be used to send segmented C_PDUs that require Expedited Delivery Service when the transmitting node needs an explicit confirmation the data was received.

A Data Transfer Sublayer entity that receives EXPEDITED DATA-ONLY (TYPE 4) D_PDU **shall** (2) send an EXPEDITED DATA-ONLY (TYPE 5) D_PDU as a selective acknowledgement of all EXPEDITED DATA-ONLY (TYPE 4) D_PDUs received from the source node.

The EXPEDITED DATA-ONLY D_PDU is similar in structure to the DATA-ONLY D_PDU. The EXPEDITED DATA-ONLY D_PDU **shall** (3) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-15 and the paragraphs noted:

- C_PDU START – **shall** (4) be as specified for the DATA-ONLY D_PDU in Section C.3.3
- C_PDU END – **shall** (5) be as specified for the DATA-ONLY D_PDU in Section C.3.3
- C_PDU ID NUMBER – **shall** (6) be as specified in the paragraphs below.

- SIZE OF SEGMENTED C_PDU– **shall** (7) be as specified in Section C.3.2.10 for all D_PDUs that have a Segmented C_PDU field.
- TRANSMIT SEQUENCE NUMBER– **shall** (8) be as specified for the DATA-ONLY D_PDU in Section C.3.3, with additional requirements as noted below.

The C_PDU ID NUMBER field **shall** (6) specify the C_PDU of which this Expedited D_PDU is a part. The value of the C_PDU ID NUMBER field **shall** (7) be an integer (modulo 16) assigned in an ascending modulo 16 order to the C_PDU, and **shall** (8) not be released for reuse with another C_PDU until the entire C_PDU has been acknowledged.

As noted above, the TRANSMIT SEQUENCE NUMBER field in the EXPEDITED DATA-ONLY D_PDU is defined and used in the same manner as that specified for the DATA-ONLY D_PDU. However, the EXPEDITED DATA-ONLY D_PDUs **shall** (9) be assigned frame numbers from a frame sequence number pool (0, 1 ...255) that is reserved exclusively for the transmission of EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs. The FRAME SEQUENCE NUMBER is used, in this D_PDU, to sequence the D_PDUs that make up a C_PDU receiving expedited delivery service.
(Note: the further implication of this requirement is that there are independent state machines and flow-control windows [or different states and sets of variables within a single state-machine] for the Expedited and Regular delivery services in the Data Transfer Sublayer).

The SEGMENTED C_PDU field is a field that is attached to the header structure defined in Figure C-15. The segmented PDU **shall** (10) immediately following the D_PDU header. Segmented C_PDUs **shall** (11) be mapped according to the convention described in C.3.2.9.

The processing of EXPEDITED D_PDUs in the EXPEDITED DATA state **shall** (11) differ from the processing of DATA-ONLY or DATA-ACK D_PDUs in the DATA state in the following ways:

- Data (i.e, C_PDUs) using the Expedited Delivery Service **shall** (12) be transferred using EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs. If duplex communication is required, EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs may be placed together in a transmission interval.

- C_PDUs requiring Expedited Delivery Service and the associated EXPEDITED D_PDUs **shall** (13) not be queued for processing within the Data Transfer Sublayer behind D_PDUs containing non-expedited data (i.e., DATA_ONLY or DATA-ACK D_PDUs).

C.3.8 EXPEDITED ACK ONLY (TYPE 5) D_PDU (Acknowledgement of simplex expedited data transfer)

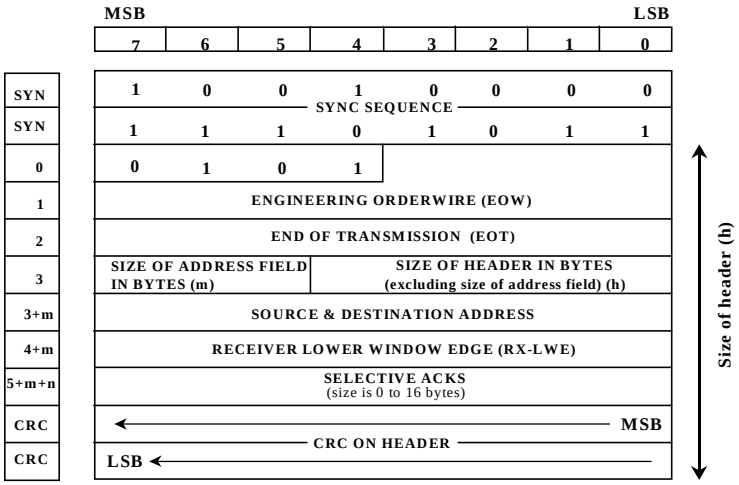


Figure C-16. Frame format for EXPEDITED ACK ONLY D_PDU Type 5

The EXPEDITED ACK-ONLY (TYPE 5) D_PDU **shall** (1) be used to selectively acknowledge received EXPEDITED DATA-ONLY D_PDUs.

The EXPEDITED ACK-ONLY (TYPE 5) D_PDU type **shall** (2) have the same format as the ACK-ONLY (TYPE 1) D_PDU, differing only in the value of the D_PDU Type field in byte 0, as specified in Figure C-16.

C.3.9 MANAGEMENT (TYPE 6) D_PDU (Management message transfer)

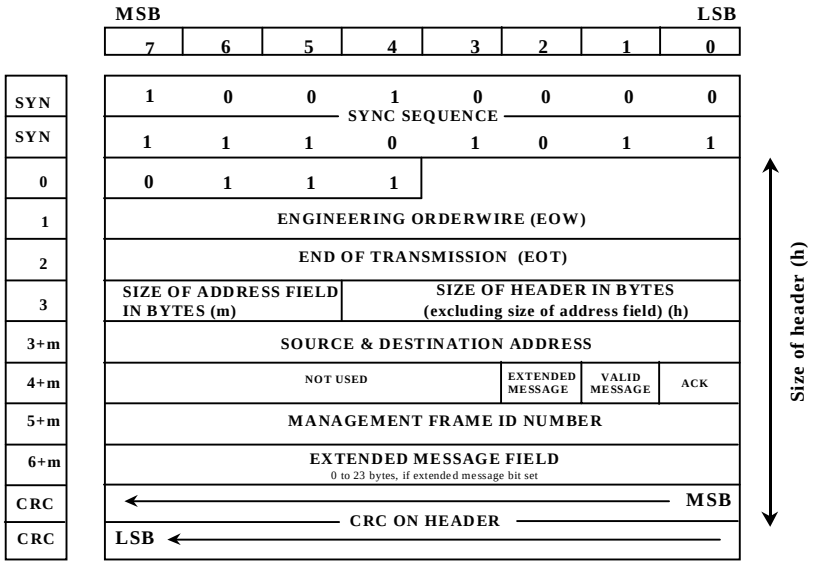


Figure C-17. Header format for MANAGEMENT D_PDU TYPE 6

The MANAGEMENT (TYPE 6) D_PDU **shall** (1) be used to send EOW Messages or Management Protocol Data Units (M_PDUs) when the transmitting node needs an explicit acknowledgement that they were received.

A Data Transfer Sublayer entity **shall** (2) acknowledge receipt of a MANAGEMENT (TYPE 6) D_PDU by sending a MANAGEMENT (TYPE 6) D_PDU with the ACK flag set to the value one (1).

The processing and transmission of MANAGEMENT (TYPE 6) D_PDUs **shall** (3) take precedence over and bypass all other pending D_PDU types in the Data Transfer Sublayer.

The exchange of MANAGEMENT D_PDUs is regulated by a stop-and-wait protocol, i.e., there **shall** (4) be only one unacknowledged MANAGEMENT D_PDU at any time.

The MANAGEMENT D_PDU **shall** (5) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-17 and the paragraphs below:

- EXTENDED MESSAGE Flag
- VALID MESSAGE
- ACK
- MANAGEMENT FRAME ID NUMBER
- EXTENDED MANAGEMENT MESSAGE

The VALID MESSAGE field **shall** (6) be set to the value one (1) if the EOW field of the D_PDU contains a valid Management message or the initial segment of a valid Management message that is continued in the EXTENDED MANAGEMENT MESSAGE field.

The VALID MESSAGE field **shall** (7) be set to the value zero (0) if the EOW field contains an Engineering Orderwire Message for which an acknowledgement message is not required

If the VALID MESSAGE field is set to zero, the MANAGEMENT D_PDU **shall** (8) be used only to acknowledge receipt of another MANAGEMENT D_PDU.

The EXTENDED MESSAGE Flag **shall** (9) be set to the value one (1) if the D_PDU contains a non-zero, non-null EXTENDED MANAGEMENT MESSAGE field.

If the EXTENDED MESSAGE Flag is set to the value zero (0), the EXTENDED MANAGEMENT MESSAGE field **shall** (10) not be present in the MANAGEMENT D_PDU.

The MANAGEMENT FRAME ID NUMBER field **shall** (11) contain an integer in the range [0,255] with which MANAGEMENT D_PDUs **shall** (12) be identified.

The Data Transfer Sublayer **shall** (13) maintain variables to manage the frame ID numbers associated with this D_PDU:

- the TX MANAGEMENT FRAME ID NUMBER **shall** (14) maintain the value of the Frame ID Number for MANAGEMENT D_PDUs that are transmitted;
- the RX MANAGEMENT FRAME ID NUMBER **shall** (15) maintain the value of the Frame ID Number for the most recently received MANAGEMENT D_PDUs.

On initialisation (such as a new connection), a node's Data Transfer Sublayer **shall** (16) set its current TX MANAGEMENT FRAME ID NUMBER to zero and **shall** (17) set its current RX MANAGEMENT FRAME ID NUMBER to an out-of-range value (i.e., a value greater than 255).

The current value of the TX MANAGEMENT FRAME ID NUMBER **shall** (18) be placed in the appropriate field of each unique MANAGEMENT D_PDU transmitted. The current value of the TX MANAGEMENT FRAME ID NUMBER **shall** (19) be incremented by one, modulo 256, after each use, unless transmission of repeated copies of the MANAGEMENT D_PDU are specified for its use, e.g., as in Section C.6.4.2.

Management D_PDUs that have been repeated (e.g., in accordance Section C.6.4.2) **shall** (20) have the same MANAGEMENT FRAME ID NUMBER.

The Data Transfer Sublayer **shall** (21) compare the MANAGEMENT FRAME ID NUMBER of received MANAGEMENT D_PDUs to the current RX MANAGEMENT FRAME ID NUMBER, and process them as follows:

- if the MANAGEMENT FRAME ID NUMBER in the received D_PDU differs from the current RX MANAGEMENT FRAME ID NUMBER value, the D_PDU **shall** (22) be treated as a new D_PDU, and the Data Transfer Sublayer **shall** (23) set the current RX MANAGEMENT FRAME ID NUMBER value equal to the value of the received MANAGEMENT FRAME ID NUMBER.

- if the value in the received D_PDU is equal to the current RX MANAGEMENT FRAME ID NUMBER value, the node **shall** (24) assume that the frame is a repetition of a MANAGEMENT D_PDU that has already been received, and the value of the current RX MANAGEMENT FRAME ID NUMBER **shall** (25) be left unchanged.

There **shall** (26) be a one-to-one correspondence between MANAGEMENT messages and MANAGEMENT D_PDUs; that is, each message is placed into a separate D_PDU (which may be repeated a number of times as specified in Section C.6.4).

The 12-bit EOW section of the D_PDU **shall** (27) carry the EOW (non-extended) MANAGEMENT message, as specified in Section C.5.

The EXTENDED MANAGEMENT MESSAGE field may be used to transmit other implementation-specific messages that are beyond the scope of this STANAG. When the EXTENDED MESSAGE field is present and in use, the EXTENDED MESSAGE Flag **shall** (28) be set to the value one (1).

C.3.10 NON-ARQ (TYPE 7) DATA D_PDU (Non-ARQ data transfer)

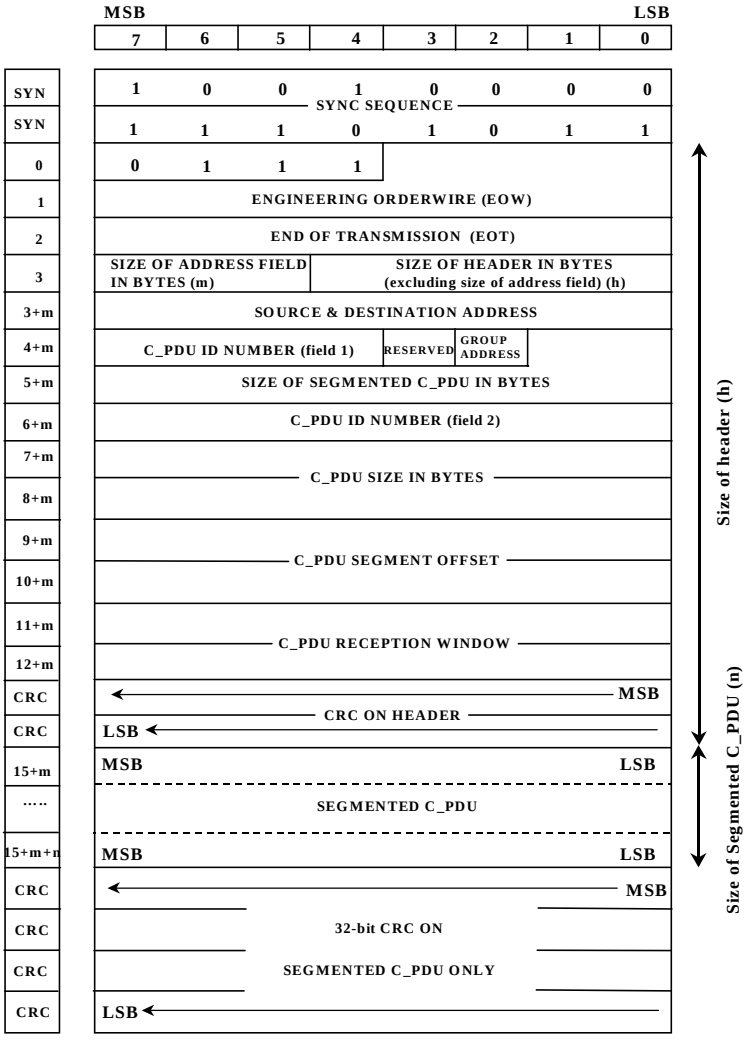


Figure C-18. Frame format for NON-ARQ DATA D_PDU TYPE 7

The NON-ARQ DATA (TYPE 7) D_PDU **shall** (1) be used to send segmented C_PDUs when the transmitting node needs no explicit confirmation the data was received.

The NON-ARQ DATA D_PDU **shall** (2) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-18 and the paragraphs below:

- C_PDU ID NUMBER (field 1)
- DELIVER IN ORDER
- GROUP ADDRESS
- SIZE OF SEGMENTED C_PDU
- C_PDU ID NUMBER (field 2)
- C_PDU SIZE
- C_PDU SEGMENT OFFSET
- C_PDU RECEPTION WINDOW

The C_PDU ID NUMBER field **shall** (3) identify the C_PDU to which the C_PDU segment

encapsulated by the NON-ARQ DATA D_PDU belongs.

The value encoded in the C_PDU ID NUMBER field **shall** (4) be a unique integer (modulo 4096) identifier assigned in an ascending order (also modulo 4096) to the C_PDU during its segmentation and encapsulation into D_PDUs.

The value encoded in the C_PDU ID NUMBER field **shall** (5) not be released for reuse and assignment to another C_PDU until the time specified in the C_PDU RECEPTION WINDOW expires, as noted below.

The C_PDU ID NUMBER space (i.e, the set of ID numbers in the range [0..4095]) for NON-ARQ DATA (TYPE 7) D_PDUs **shall** (6) be different than the similarly-defined number space for EXPEDITED NON-ARQ DATA (TYPE 8) D_PDUs.

The value of the C_PDU ID NUMBER **shall** (7) be encoded in a 12 bit field as specified in Figure C-19.

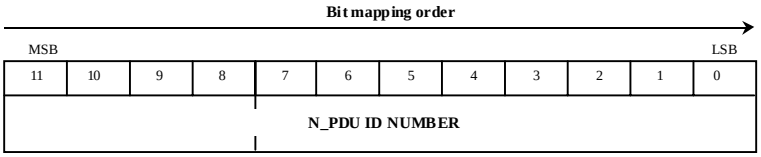


Figure C-19. C_PDU ID NUMBER Field

The value of the C_PDU ID NUMBER **shall** (8) be mapped into the NON-ARQ DATA D_PDU into two split fields as follows, and as depicted in Figure C-20:

- The four-most-significant bits of the value of the C_PDU ID NUMBER **shall** (9) be mapped into C_PDU ID NUMBER (field 1);
- The eight least-significant bits of the value of the C_PDU ID NUMBER **shall** (10) be mapped into C_PDU ID NUMBER (field 2).

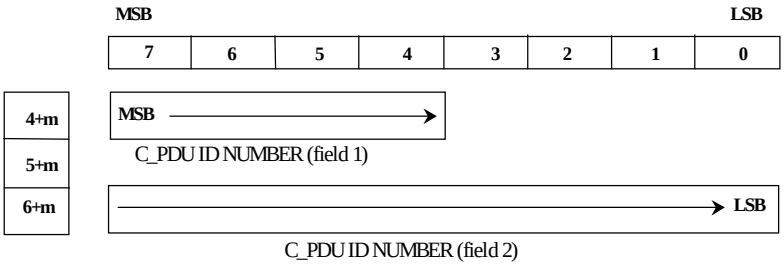


Figure C-20. C_PDU ID NUMBER mapping convention in D_PDU header

If the DELIVER IN ORDER flag is set (i.e., its value equals 1) on the D_PDUs composing a C_PDU, the C_PDU **shall** (11) be delivered to the network layer when both the following conditions are met:

1) C_PDU is complete and error free or the C_PDU RECEPTION WINDOW expires.
2) Previous C_PDUs that also had the DELIVER IN ORDER flag set have been delivered.

If the DELIVER IN ORDER flag is cleared (0) on the D_PDUs composing a C_PDU, the C_PDU **shall** (12) be delivered to the network layer when the following condition is met.

1) C_PDU is complete and error free or the C_PDU RECEPTION WINDOW expires.

The GROUP ADDRESS flag **shall** (13) indicate that the destination address should be interpreted as a group address rather than an individual address, as follows:

- The destination address **shall** (14) be interpreted as a group address when the GROUP ADDRESS flag is set (1).
- However when the GROUP ADDRESS flag is cleared (0) the destination address **shall** (15) be interpreted as an individual node address.

Note: If a specific PDU is intended for only one node, the node's normal address may also be used with the NON-ARQ DATA D_PDU. Group addresses are intended to allow PDUs to be addressed to specific groups of nodes when using NON-ARQ DATA D_PDUs. The use of a bit to designate a "group address" allows the same number (~268 million!) and structure of group addresses to be designated as for normal addresses, rather than requiring some portion of the total address space for group addresses. The management of group addresses is outside of the scope of this STANAG.

The SIZE OF SEGMENTED C_PDU field **shall** (16) specify the number of bytes contained in the SEGMENTED C_PDU file in accordance with the requirements of Section C.2.10.

The C_PDU SIZE field **shall** (17) indicate the size in bytes of the C_PDU of which the C_PDU segment encapsulated in this D_PDU is a part.

The value of the C_PDU SIZE field **shall** (18) be encoded in a 16 bit field, with the bits mapped as specified by Figure C-21. The number **shall** (19) be mapped into the D_PDU by placing the MSB of the field into the MSB of the first byte in the D_PDU as can be seen in Figure C-22.

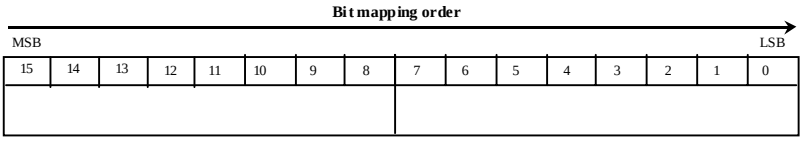


Figure C-21. C_PDU SIZE Field

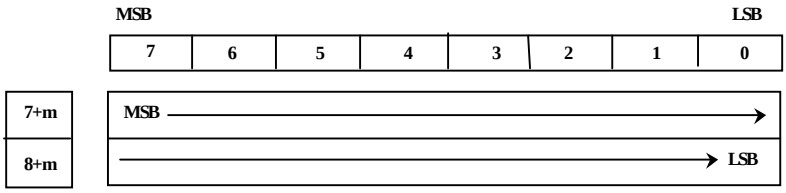


Figure C-22. C_PDU SIZE mapping convention in D_PDU header

The C_PDU SEGMENT OFFSET field **shall** (20) indicates the location of the first byte of the SEGMENTED C_PDU with respect to the start of the C_PDU. For the purposes of this field,

the bytes of the C_PDU **shall** (21) be numbered consecutively starting with 0.

The C_PDU SEGMENT OFFSET field is a 16 bit field, the bits **shall** (22) be mapped as specified by Figure C-23. The number **shall** (23) be mapped into the D_PDU by placing the MSB of the field into the MSB of the first byte in the D_PDU as specified in Figure C-24.

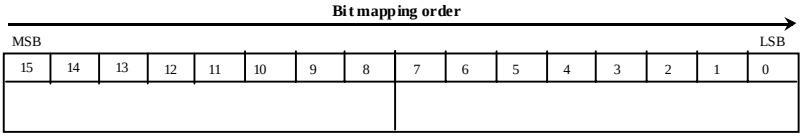


Figure C-23. C_PDU SEGMENT OFFSET Field

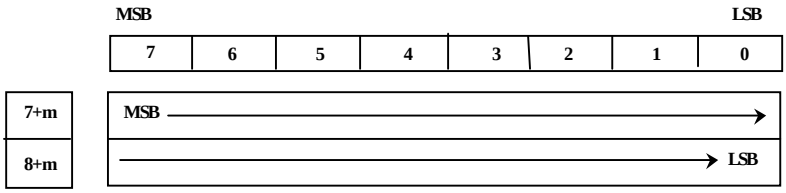


Figure C-24. C_PDU SEGMENT OFFSET mapping convention in D_PDU header

The C_PDU RECEPTION WINDOW field **shall** (24) indicate the maximum remaining time in units of half (1/2) seconds relative to the start of the D_PDU during which portions of the associated C_PDU may be received.

As in the case of the EOT field, the C_PDU RECEPTION WINDOW **shall** (25) be updated just prior to transmitting each D_PDU. The receiving node can use this information to determine when to release a partially received C_PDU. (The transmitter is not allowed to transmit D_PDUs that are a portion of a C_PDU when the C_PDU RECEPTION WINDOW has expired).

The value of the C_PDU RECEPTION WINDOW field **shall** (26) be encoded in a 16 bit field with the bits be mapped as specified by Figure C-25. The value **shall** (27) be mapped into the D_PDU by placing the MSB of the field into the MSB of the first byte in the D_PDU as specified in Figure C-26.

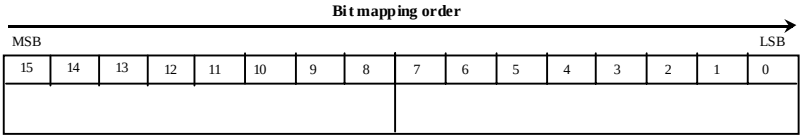


Figure C-25. C_PDU RECEPTION WINDOW Field

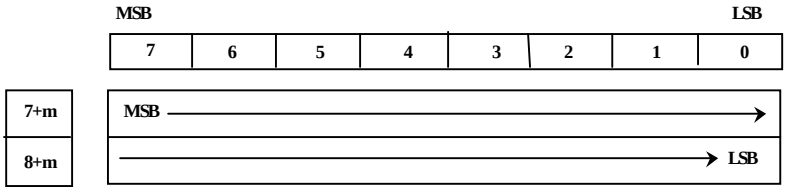


Figure C-26. C_PDU RECEPTION WINDOW mapping convention in D_PDU header

C.3.11 EXPEDITED NON-ARQ DATA (TYPE 8) D_PDU (Expedited non-ARQ data transfer)

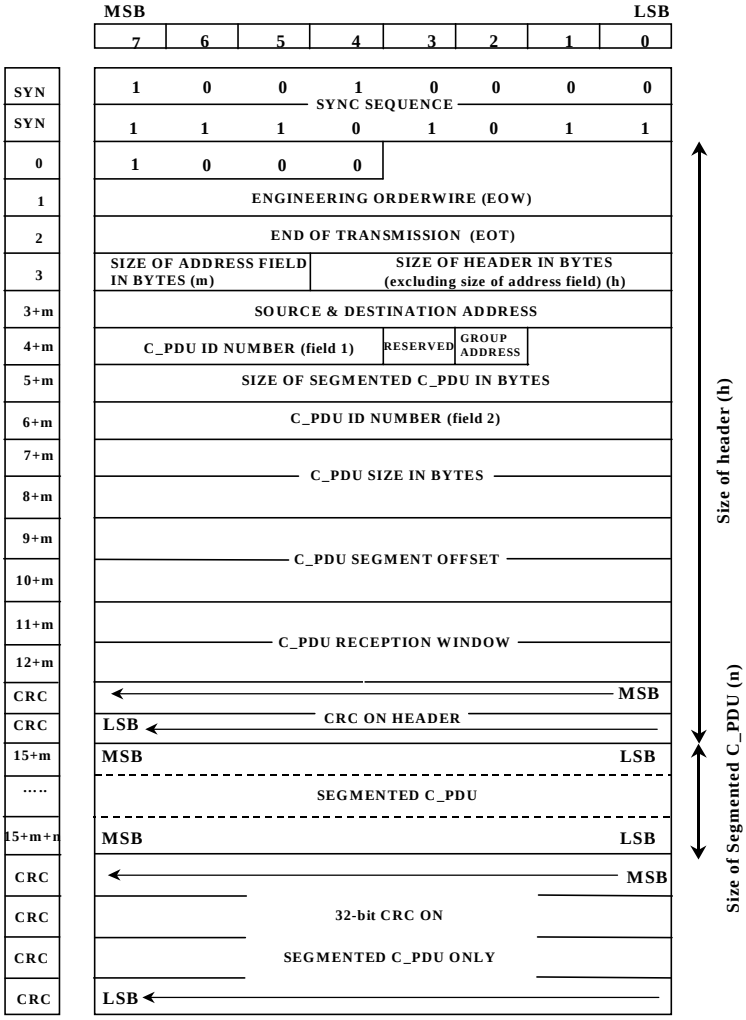


Figure C-27. Frame format for EXPEDITED NON-ARQ DATA D_PDU Type 8

The frame format for EXPEDITED NON-ARQ DATA (TYPE 8) D_PDUs **shall** (1) be identical to the NON_ARQ DATA D_PDU with the exception that the TYPE field has a value of 8, as specified in Figure C-27.

The C_PDU ID NUMBER space (i.e, the set of ID numbers in the range [0..4095]) for EXPEDITED NON-ARQ DATA (TYPE 8) D_PDUs **shall** (2) be different than the similarly-defined number space for NON-ARQ DATA (TYPE 7) D_PDUs.

C.3.12 WARNING (TYPE 15) D_PDU

(in response to unexpected or unrecognised D_PDU type)

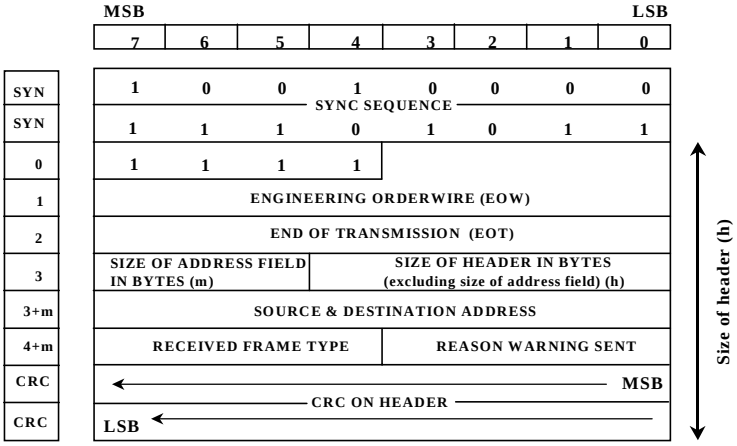


Figure C-28. Frame format for WARNING D_PDU Type 15

A Data Transfer Sublayer **shall** (1) a WARNING D_PDU to any remote node from which an unexpected or unknown D_PDU type has been received.

The WARNING D_PDU **shall** (2) contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-28 and the paragraphs below:

- RECEIVED FRAME TYPE
- REASON WARING SENT

The RECEIVED FRAME TYPE field **shall** (3) indicate the frame type that caused the warning to be sent.

The value of the RECEIVED FRAME TYPE field **shall** (4) be encoded in four bits, and located within the D_PDU as specified in Figure C-29 and Figure C-30.

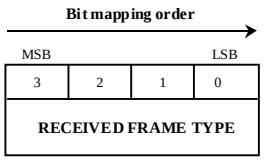


Figure C-29. RECEIVED FRAME TYPE Field

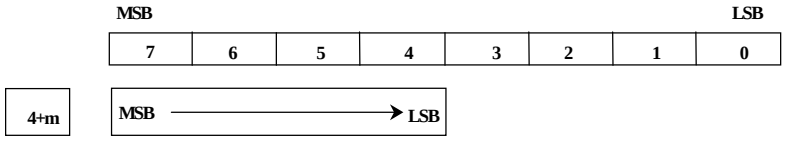


Figure C-30. RECEIVED FRAME TYPE mapping convention in D_PDU header

The REASON WARNING SENT field **shall** (5) indicate the reason the frame type caused a

warning, with values as Specified in Table C-3:

Table C-3: Encoding of WARNING D_PDU Reason Field

Reason	Field Value
Unrecognised D_PDU type Received	0
Connection-related D_PDU Received While Not Currently Connected	1
Invalid D_PDU Received	2
Invalid D_PDU Received for Current State	3
Unspecified/reserved	4-15

The value of the REASON WARNING SENT field **shall** (6) be encoded in four bits, and located within the D_PDU as specified in Figure C-31 and Figure C-32.

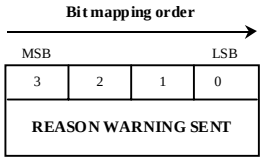


Figure C-31. REASON WARNING SENT Field

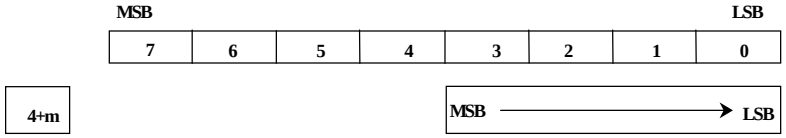


Figure C-32. REASON WARNING SENT mapping convention in D_PDU header

The transmission of WARNING type D_PDUs **shall** (7) be initiated independently by the Data Transfer Sublayer in response to certain D_PDUs and **shall** (8) not be acknowledged explicitly.

WARNING type D_PDUs may be inserted into an ongoing transmission interval provided that the transmission interval period is not exceeded.

A WARNING D_PDU **shall** (9) be sent in the following conditions:

1. A node receives a D_PDU header addressed to itself with a valid CRC and an unrecognised D_PDU type (value 0000)
2. A node is not in the IDLE/BROADCAST state and it receives a D_PDU header addressed to itself, from a node with which it is not currently connected. (value 0001)
3. A node is in IDLE/BROADCAST state and it receives a D_PDU header addressed to itself which is other than type 7 or type 8 D_PDU (value 0010)
4. A node receives any D_PDU which is recognized but is not of the allowed type for the state which the receiving node is in (value 0011; this is the general case of the preceding)

A WARNING D_PDU **shall** (10) not be sent in response to receipt of a WARNING D_PDU.

C.4 C_PDU Segmentation and Re-assembly Processes

The process of C_PDU segmentation and re-assembly **shall** (1) be as defined in the subsections that follow for ARQ and non-ARQ delivery services provided to regular and expedited C_PDUs.

C.4.1 ARQ Mode Segmentation and Re-assembly

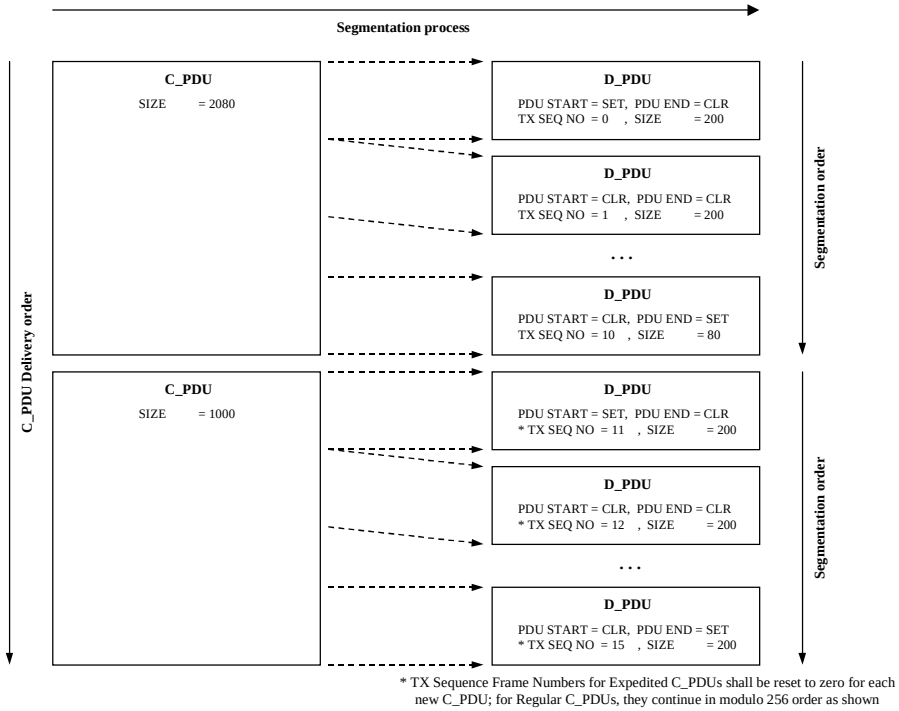


Figure C-33. C_PDU Segmentation for ARQ Delivery Services (Regular and Expedited Service)

Segmentation of a C_PDU into segments small enough to fit within a D_PDU for ARQ-delivery (i.e, a DATA, DATA-ACK, or EXPEDITED-DATA D_PDU) **shall** (1) be performed in accordance with the example shown in Figure C-33 and as follows:

a. The Maximum C_PDU-Segment Size within a D_PDU for ARQ-Delivery services **shall** (2) be a configurable parameter no greater than 1023 bytes in any implementation compliant with this STANAG. An implementation may configure the Maximum C_PDU-Segment Size to match the interleaver size for optimum channel efficiency or other reasons.

b. An entire C_PDU that is smaller than the Maximum C_PDU-Segment Size **shall** (3) be placed in the C_PDU segment of a single D_PDU.

c. A DATA or DATA-ACK, or EXPEDITED D_PDU that contains an entire C_DPU **shall** (4) be marked with the both C_PDU START field and the C_PDU END field set equal to the value “1”. [Note: An ‘only’ C_PDU segment is both the “first” and “last” segment of a sequence of one.]

d. The Data Transfer Sublayer **shall** (5) divide C_PDUs larger than the Maximum C_PDU-Segment Size into segments that are no larger than the Maximum C_PDU Segment Size.

e. Only the last segment or the only segment taken from a C_PDU may be smaller than the Maximum C_PDU-Segment size. A C_PDU smaller than the Maximum C_PDU-Segment size **shall** (6) be placed only in the D_PDU that contains the last segment of the C_PDU, i.e., only in a D_PDU for which the C_PDU END field is set equal to one.

f. The bytes within a C_PDU segment **shall** (7) be taken from the source as a contiguous sequence of bytes in the same order in which they occurred in the source C_PDU.

g. D_PDUs containing C_PDU segments taken in sequence from the source C_PDU **shall** (8) have sequential Frame Sequence number fields, modulo 256. [Note: With the first C_PDU segment placed in a D_PDU with Frame Sequence = P, the second would have Frame Sequence = P+1, the third P+2, and so on, with Frame-Sequence operations performed modulo-256.]

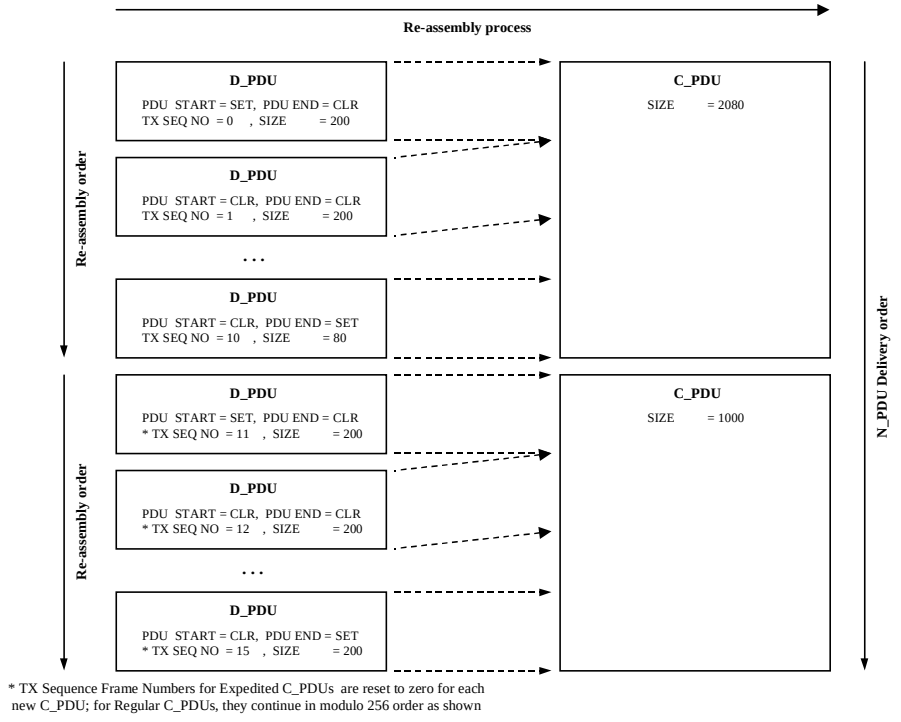


Figure C-34. C_PDU Re-assembly for ARQ Delivery Services (Regular and Expedited Service)

Re-assembly of a C_PDU from its segments **shall** (9) be performed in accordance with the example shown in Figure C-34 and as follows (unless noted otherwise, C_PDU segments that are reassembled are expected to have passed the CRC error-check and have no detectable errors):

(a) The re-assembly process for C_PDUs receiving ARQ-service **shall** (9) use the Frame-Sequence-Number field, C_PDU START flag, and C_PDU END flag to determine when all segments of a C_PDU have been received.

(b) A C_PDU segment taken from a D_PDU whose C_PDU START and C_PDU END flags are both set to the value "one" (1) **shall** (10) be taken as a single C_PDU and processed as follows:

1) If the D_PDU is for a regular (unexpedited) data type, the C_PDU **shall** (11) be delivered to the Channel Access Sublayer using a D_UNIDATA_INDICATION primitive;

2) If the D_PDU is for an unexpedited data type, the C_PDU **shall**(12) be delivered to the Channel Access Sublayer using a D_EXPEDITED_UNIDATA_INDICATION primitive;

(c) A segment from a C_PDU larger than the Maximum C_PDU Segment Size **shall** (13) be combined in modulo 256 order with other segments whose D_PDU Frame Sequence Numbers lie in the range defined by the Frame Sequence Numbers of the C_DPU START and C_PDU END segments;

(d) A completely reassembled C_PDU **shall** (14) be delivered to the Channel Access Sublayer using the appropriate D_Primitive.

C.4.2 NON-ARQ Mode Segmentation and Re-assembly

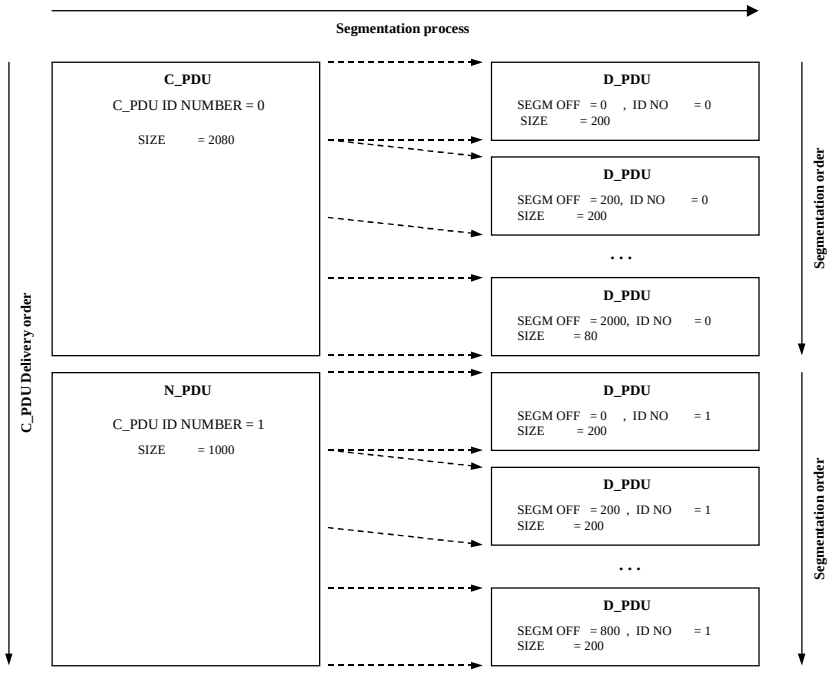


Figure C-35. C_PDU Segmentation for Non-ARQ Delivery Services (Regular and Expedited Service)

Segmentation of a C_PDU into segments small enough to fit within a D_PDU for non-ARQ-delivery (i.e, a Non-ARQ DATA, or EXPEDITED-Non-ARQ-DATA D_PDU) **shall** (1) be performed in accordance with the example shown in Figure C-35, and as follows:

a. The Maximum C_PDU-Segment Size within a D_PDU for non-ARQ-Delivery services **shall** (2) be a configurable parameter no greater than 1023 bytes in any implementation compliant with this STANAG. An implementation may configure the Maximum C_PDU-Segment Size to match the interleaver size for optimum channel efficiency or other reasons;

b. An entire C_PDU for non-ARQ delivery that is smaller than the Maximum C_PDU-Segment Size **shall** (3) be placed in the C_PDU segment of a single D_PDU;

c. A unique C_PDU ID number **shall** (4) be assigned to the non-ARQ C_PDU in accordance with the requirements of Section C.3.10;

d. all D_PDUs containing segments from the same C_PDU **shall** (5) have the same C_PDU ID number;

e. The Segment Offset field of the D_PDU containing the first segment from a C_PDU **shall** (6) be equal to zero;

f. The Segment Offset field of the D_PDU containing any subsequent segment from a C_PDU **shall** (7) be set equal to the number of bytes from the original C_PDU that precede the first byte of the segment.

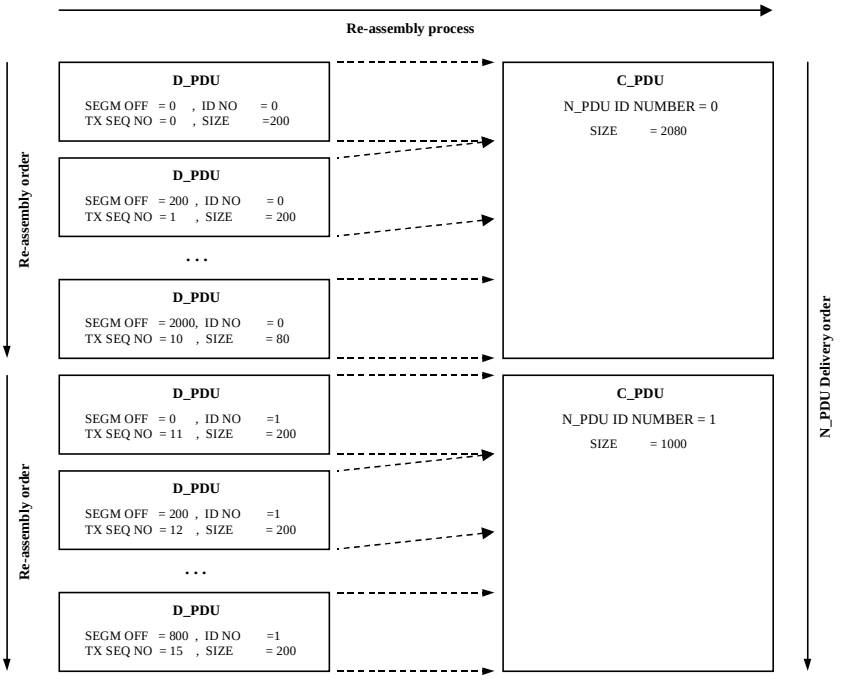


Figure C-36. C_PDU Re-assembly for Non-ARQ Delivery Services (Regular and Expedited Service)

For Non-ARQ services, re-assembly of a C_PDU from its segments **shall** (8) be performed in accordance with the example shown in Figure C-36 and as follows (unless noted otherwise, C_PDU segments that are reassembled are expected to have passed the CRC error-check and have no detectable errors):

a) The re-assembly process for Non-ARQ C_PDUs **shall** (9) use the C_PDU ID Number, Segment-Offset field, C_PDU-Segment-Size field, and C_PDU-Size field to determine when all segments of a C_PDU have been received.

b) If the Error-Free Delivery Mode has been specified, a reassembled C_PDU **shall**(10) be delivered if and only if all segments of the C_PDU have been received without errors;

c) If the Deliver-w/-Errors Mode has been specified, the re-assembly process **shall** (11) proceed as follows:
   1) C_PDU segments received without detected errors **shall** (12) be collected as received in their D_PDUs and placed in order within the reassembled C_PDU;
   2) C_PDU segments received with detected errors **shall** (13) be placed within the reassembled C_PDU just as they are received in their D_PDUs (i.e, with errors), with the size in bytes and the position of the first byte of the segment noted in the D_Primitive used to deliver the C_PDU to the Channel Access Sublayer;
   3) At the end of a specified and configurable timeout-interval the size in bytes and the position of the first byte of any C_PDU segments that have been lost or still not received **shall** (14) be noted in the D_Primitive that delivers the C_PDU to the Channel Access Sublayer

d) If the Deliver-In-Order Mode has been specified (with or without the Deliver-with-Errors Mode specified), C_PDUs **shall** (15) be delivered to the Channel Access Sublayer only if C_PDUs with lower-numbered C_PDU ID Numbers have already been delivered.

e) If the Deliver-out-of-Order Mode has been specified, C_PDUs **shall** (16) be delivered to the Channel Access Sublayer as soon as all segments have been received (in Error-Free Mode) or received and accounted for (in Deliver-with-Errors Mode).

f) Delivery of the reassembled D_PDU **shall** (17) be performed with the D_Primitive appropriate for the type of data (i.e., regular or expedited) received.

C.5 EOW and Management Message Types

The basic set of EOW Messages may be placed in the 12-bit EOW section of any D_PDU for which an explicit acknowledgement of the EOW Message is not required, and in the MANAGEMENT D_PDU whenever an explicit acknowledgement of the EOW Message is required. The internal structure and use of these messages is specified in this section.

In addition, Extended Management messages may be defined as implementation-specific extensions to the STANAG in accordance with the format and use of the MANAGEMENT D_PDU. The specification and use of Extended Management messages is beyond the scope of this STANAG. Implementation-specific Extended Management messages **shall** (1) be identified and encoded through the use of the Extended Message Flag as specified in Section C.3.9 for the Management D_DPU.

The types of EOW messages **shall** (2) be as defined in the table below:

Table C-4. EOW Message Types

Message Type	Function	Contents
0	RESERVED	all bits reset to 0
1	Data Rate Change Request (DRC_Req)	New HF modem transmit data rate and interleaving setting for DRC master
2	Data Rate Change Response (DRC_Resp)	Positive or negative response (including reason if negative)
3	Unrecognized Type Error: user defined message type	message type field which is not recognized
4	Capability Advertisement	bit map describing capabilities of node
5	Frequency Change/ ALM Request	As defined in Annex I – Messages and Procedures for Frequency Change
6	Frequency Change/ ALM Response	As defined in Annex I - Messages and Procedures for Frequency Change
7-15	*Unspecified/user-defined*	

The format of the EOW message types **shall** (3) be as shown in Figure C-37.

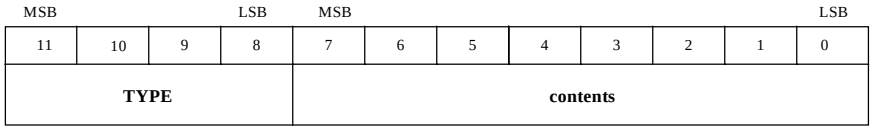


Figure C-37. Generic Format of EOW Messages

The TYPE field of the EOW message **shall** (4) be filled with the hexadecimal value of the appropriate message type (units only), with the LSB of the TYPE value placed in the LSB of the TYPE field.

The Contents field **shall** (5) be EOW type-specific, in accordance with the subsections below.

C.5.1 Data Rate Change Request (TYPE 1) EOW Message

MSB			LSB	MSB			LSB	MSB	LSB	MSB	LSB
11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**1**	**Data Rate**				**Interleaving**		**Other parameters**	
TYPE											

Figure C-38. Format of Management Message Type 1.

The Data Rate Change Request (TYPE 1) EOW Message **shall** (1) be used in conjunction with the Data Rate Change protocol, as specified in Section C.6.4.

The Data Rate Change Request (TYPE 1) EOW Message may be used by a receiving node as an advisory message indicating the modem parameters to which the link may be changed to improve performance. Alternatively, a transmitter may use the Data Rate Change Request (TYPE 1) EOW Message to request a receiver to change to the indicated parameters.

The Data Rate Change Request (TYPE 1) EOW Message **shall** (2) be formatted and encoded as specified in Figure C-38 and the paragraphs that follow, and includes the following type-specific subfields:

- Data Rate
- Interleaving
- Other Parameters

The Data Rate parameter **shall** (3) be the rate at which the node originating the message (i.e, either the DRC Master or Advisee, as noted in Section C.6.4 specifying Data Rate Change Procedures) wishes to transmit data, in accordance with the encoding defined in the following table:

Table C-5. Data Rate Parameter Message Type 1

MSB - LSB	Interpretation
0 0 0 0	75 bps
0 0 0 1	150 bps
0 0 1 0	300 bps
0 0 1 1	600 bps
0 1 0 0	1200 bps
0 1 0 1	2400 bps
0 1 1 0	3200 bps
0 1 1 1	3600 bps
1 0 0 0	4800 bps
1 0 0 1	6400 bps
1 0 1 0	7200 bps
1 0 1 1	9600 bps
others	reserved

The Interleaver Parameter field **shall** (4) specify the interleaver requested for use by the node producing the message (for that link, if multiple links/modems are in use) with respect to transmit and receive operation, in accordance with the following table:

Table C-6. Interleaver Parameter: Message Type 1

MSB - LSB	Interpretation
0 0	no interleaving
0 1	short interleaving
1 0	long interleaving
1 1	reserved

The Other Parameters field **shall** (5) specify the capabilities of the modem in use by the node producing the message (for that link, if multiple links/modems are in use) with respect to transmit and receive data rates, and whether the message is an advisory message or request message, in accordance with the following table:

Table C-7. Contents for Type 1 Message (Other Parameters)

MSB - LSB	Interpretation
0 0	DRC Request: master has independent data rate (change applies to Tx data rate only)
0 1	DRC Request: Tx and Rx data rate at master must be equal (change will apply to both Tx and Rx data rates)
1 0	DRC Advisory: Advising node has independent data rate for Tx and Rx (change applies to Rx data rate only)
1 1	DRC Advisory: Tx and Rx data rate at advising node must be equal (change will apply to both Tx and Rx data rates)

C.5.2 Data Rate Change Response (TYPE 2) EOW Message

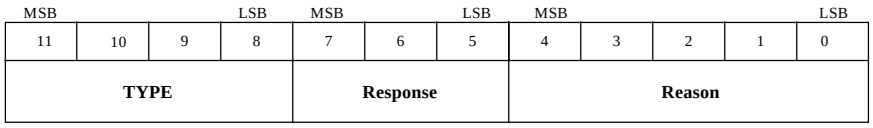


Figure C-39. Format of Type 2 Message

The Data Rate Change Response (TYPE 2) EOW Message **shall** (1) be used in conjunction with the Data Rate Change protocol, as specified in Section C.6.4.

The Data Rate Change Response (TYPE 2) EOW Message **shall** (2) be encoded as shown in Figure C-39, and include the following fields:

- Response
- Reason

The Response field **shall** (3) indicate the originator's response to the last DRC-related message it received, with possible responses and their encoding as defined in the following table:

Table C-7. Contents for Type 2 Message (Response)

MSB - LSB	Interpretation
0 0 0	accept
0 0 1	refuse
0 1 0	cancel
0 1 1	confirm

The Reason field **shall** (4) indicate the originator's reason for its response, with possible reasons and their encoding as defined in the following table:

Table C-8. Contents for Type 2 Message (Reason)

MSB - LSB	Interpretation
0 0 0 0 0	no reason (used to indicate unconditional acceptance of DRC_Request)
0 0 0 0 1	Tx and Rx parameters must be the same (conditionally accept)
0 0 0 1 0	Not possible to change modem data rate
0 0 0 1 1	Not possible to change modem interleaving
0 0 1 0 0	Not possible to change modem data rate or interleaving
0 0 1 0 1	Not consistent with local conditions

C.5.3 Unrecognised Type Error (TYPE 3) EOW Message

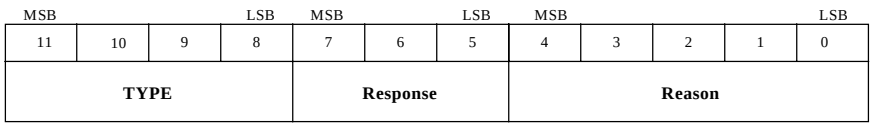


Figure C-40. Format of Type 3 Message

The UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message **shall** (1) be used to declare an error related to receipt of EOW message.

The UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message **shall** (2) be encoded as shown in Figure C.40 and include the following fields:

- Response
- Reason

The type of the unrecognised EOW message or the message that triggered the error **shall** (3) be mapped into the four least-significant bits of the "reason" field.

The unused MSB of the "reason" field, and all bits in the "response" field, **shall** (4) be reset to the value zero (0).

C.5.4 Capability Advertisement (TYPE 4) EOW Message

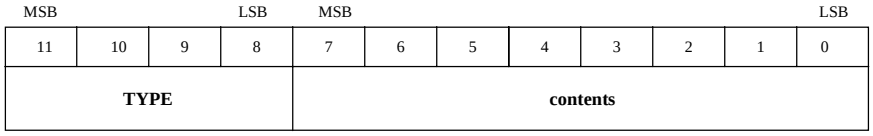


Figure C-41. Format of Capabilities (Type 4) MANAGEMENT Messages

The EOW Message Type 4 allows a node to inform another node of its capabilities related to modem parameters, waveform, and control.

The EOW Message Type 4 **shall** (1) be encoded as shown in Figure C-41 and contains a single field, Content.

The Content field **shall** (2) be encoded as the bit-mapped specification of capabilities defined in the following table:

Table C-9. Contents of Message field for MANAGEMENT Message Type 4

bit	meaning
7 (MSB)	Adaptive modem parameters (DRC) capable note 1 (0 = no, 1 = yes)
6	STANAG 4529 available note 2 (0 = no, 1 = yes)
5	MIL-STD-188-110A available note 2 (0 = no, 1 = yes)
4	extended data rate capable note 3 (0 = no, 1 = yes)
3	full duplex supported note 4 (0 = no, 1 = yes)
2	split frequency supported note 4 (0 = no, 1 = yes)
1	non-ARCS ALE capable
0 (LSB)	ARCS capable (0 = no, 1 = yes)

Notes

1. If a node is DRC capable, it must implement at minimum 75 bps through 2400 bps (1200 bps for STANAG 4529) with short and long interleaving in accordance with the relevant document.
2. All nodes shall have, at minimum, the STANAG 4285 waveform available.
3. Annex G describes waveforms for data rates above 2400 bps.
4. A full duplex node must have split frequency operation available.

C.6 Peer-to-peer Communication Protocols

This section discusses the interactions between the Data Transfer Sublayer entities at different nodes in terms of states, state-transition diagrams, and state tables for a hypothetical state machine. This STANAG does not mandate a state machine implementation. The requirement for interoperability is that the system act consistently with the state-transition and actions rules for message exchange and format presented in this STANAG.

C.6.1 Data Transfer Sublayer States and Transitions

The expected and allowed interactions of one node with another are described herein with respect to states of the node's Data Transfer sublayer. Receiving certain PDUs (from the modem) or D_Primitives (from the Channel Access Sublayer) will cause a node, depending on its state, to transmit certain PDUs and/or D_Primitives, and/or change to another state.

The Data Transfer Sublayer interactions with peers **shall** (1) be defined with respect to the states shown in Figure C-42 and as follows:

IDLE(UNCONNECTED)	IDLE(CONNECTED)
DATA(UNCONNECTED)	DATA(CONNECTED)
EXPEDITED-DATA(UNCONNECTED	EXPEDITED-DATA(CONNECTED)
MANAGEMENT(UNCONNECTED)	MANAGEMENT(CONNECTED)

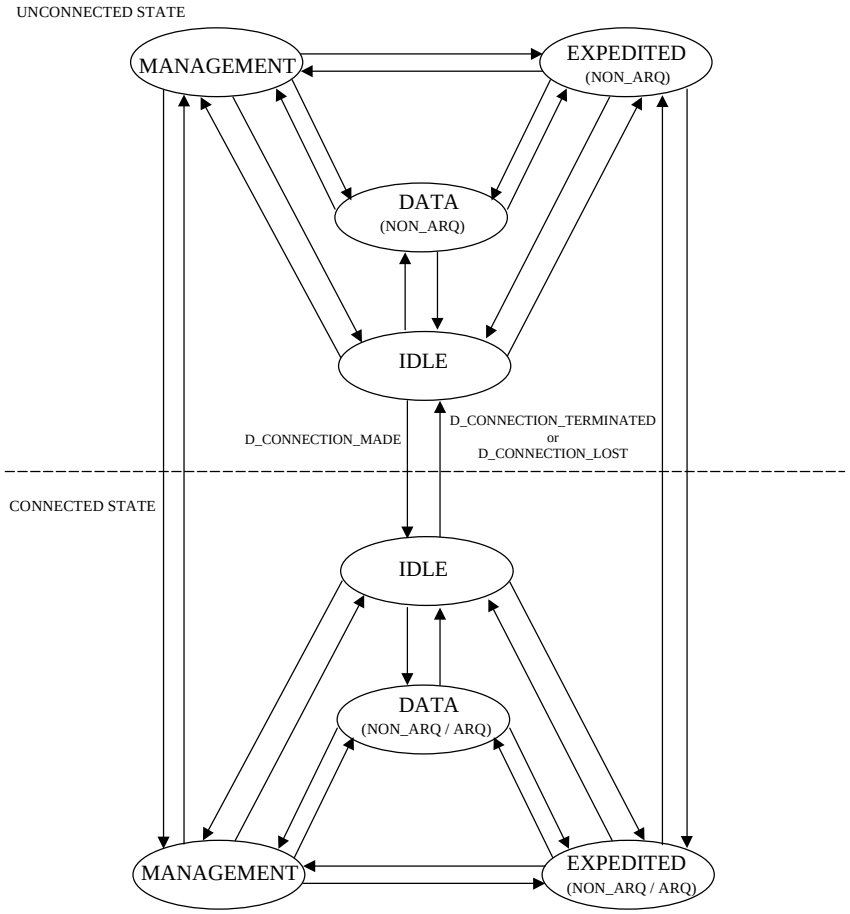


Figure C-42. Nominal State and Transition Diagram (Data Transfer Sublayer)

C.6.1.1 State Transition Rules

The transitions between DTS States and the actions that arise from given events **shall** (1) be as defined in the tables presented in the subsections that follow.

DTS states, transitions, and actions **shall** (2) be defined and maintained with respect to a specified remote node address.

In all of the tables below, "PDU received" refers to a PDU received that is addressed to the node in question from the specified remote node for which the states are maintained. Action and transitions rules **shall** (3) not occur based on PDUs addressed to other nodes or from a node other than the specified remote node for which the states are maintained.

Likewise, "D_Primitive received" refers to a D_Primitive received from the Channel Access sublayer that references the specified remote node. Action and transitions rules **shall** (4) not occur based on D_Primitives that reference nodes other than the specified remote node.

DTS states **shall** (5) be maintained for each specified node for which a connection or ARQ protocol must be maintained. [Note: If multiple links, as defined by the Channel Access or Subnetwork Interface sublayers, must be maintained simultaneously, then a set of DTS State Variables must be maintained for each of the links. If links are not maintained simultaneously, DTS State variables may be reused for any node]

C.6.1.1.1 IDLE(UNCONNECTED) State Transition Rules

When in the IDLE(UNCONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-10. Event/Transition/Action Rules for IDLE (UNCONNECTED) State: CAS-Related Events

Received D_Primitive/ Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.]
D_CONNECTION_ TERMINATED	IDLE(UNCONNECTED) , i.e., No Change.	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.]
D_UNIDATA_REQUEST (ARQ)	IDLE(UNCONNECTED) , i.e., No Change.	REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	IDLE(UNCONNECTED) , i.e., No Change.	REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_UNIDATA_REQUEST (NON-ARQ)	DATA(UNCONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	EXPEDITED DATA (UNCONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs

When in the IDLE(UNCONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-11. Event/Transition/Action Rules for IDLE (UNCONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
ACK Type 1 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
DATA-ACK Type 2 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
RESET/WIN RESYNC Type 3 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-DATA Type 4 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-ACK Type 5 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (UNCONNECTED)	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	DATA(UNCONNECTED)	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	EXPEDITED-DATA (UNCONNECTED)	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.2 DATA(UNCONNECTED) State Transition Rules

When in the DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-12. Event/Transition/Action Rules for DATA (UNCONNECTED) State: CAS-Related Events

Received D_Primitive Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.]
D_CONNECTION_ TERMINATED	DATA (UNCONNECTED) , i.e., No Change.	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.]
D_UNIDATA_REQUEST (ARQ)	DATA (UNCONNECTED) , i.e., No Change.	REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	DATA(UNCONNECTED) , i.e., No Change.	REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_UNIDATA_REQUEST (NON-ARQ)	DATA(UNCONNECTED) , i.e., No Change.	SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	EXPEDITED-DATA (UNCONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs

When in the DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-13. Event/Transition/Action Rules for DATA (UNCONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	DATA(UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
ACK Type 1 D_PDU	DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
DATA-ACK Type 2 D_PDU	DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
RESET/WIN RESYNC Type 3 D_PDU	DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-DATA Type 4 D_PDU	DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-ACK Type 5 D_PDU	DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (UNCONNECTED)	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	DATA(UNCONNECTED), i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	DATA(UNCONNECTED), i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_ INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	DATA(UNCONNECTED), i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.3 EXPEDITED-DATA(UNCONNECTED) State Transition Rules

When in the EXPEDITED-DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall**(1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-14. Event/Transition/Action Rules for EXPEDITED-DATA (UNCONNECTED) State: CAS-Related Event

Received D_Primitive Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.]
D_CONNECTION_ TERMINATED	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.]
D_UNIDATA_REQUEST (ARQ)	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_UNIDATA_REQUEST (NON-ARQ)	IF completed transmitting all Expedited-Data, change to DATA(UNCONNECTED), OTHERWISE, EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	IF completed sending all Expedited-Data, SEGMENT C_PDU, then, COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE, QUEUE C_PDU Pending completion of Expedited-Data transmission.
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs

When in the EXPEDITED-DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall**(2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-15. Event/Transition/Action Rules for EXPEDITED-DATA (UNCONNECTED) State: Reception-Related Event

Received D_PDU/ Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
ACK Type 1 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
DATA-ACK Type 2 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
RESET/WIN RESYNC Type 3 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-DATA Type 4 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-ACK Type 5 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (UNCONNECTED)	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.4 MANAGEMENT(UNCONNECTED) State Transition Rules

When in the MANAGEMENT(UNCONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-16. Event/Transition/Action Rules for MANAGEMENT (UNCONNECTED) State: CAS-Related Events

Received D_Primitive/ Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	MANAGEMENT (CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.]
D_CONNECTION_ TERMINATED	MANAGEMENT (UNCONNECTED) , i.e., No Change.	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.]
D_UNIDATA_REQUEST (ARQ)	MANAGEMENT (UNCONNECTED) , i.e., No Change.	REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	MANAGEMENT (UNCONNECTED) , i.e., No Change.	REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.]
D_UNIDATA_REQUEST (NON-ARQ)	IF Management Protocol completed, DATA(UNCONNECTED), OTHERWISE, MANAGEMENT (UNCONNECTED) , i.e., No Change.	IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol.
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	IF Management Protocol completed, EXPEDITED DATA (UNCONNECTED), OTHERWISE, MANAGEMENT (UNCONNECTED) , i.e., No Change.	IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol.

When in the MANAGEMENT(UNCONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-17. Event/Transition/Action Rules for MANAGEMENT (UNCONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
ACK Type 1 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
DATA-ACK Type 2 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
RESET/WIN RESYNC Type 3 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-DATA Type 4 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
EXPEDITED-ACK Type 5 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.5 IDLE(CONNECTED) State Transition Rules

When in the IDLE(CONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-18. Event/Transition/Action Rules for IDLE (CONNECTED) State: CAS-Related Event

Received D_Primitive Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED), i.e. No Change	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.]
D_CONNECTION_ TERMINATED	IDLE(UNCONNECTED)	TERMINATE ARQ processing; RESET State Variables.
D_UNIDATA_REQUEST (ARQ)	DATA(CONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant ARQ DATA Type 0 D_PDUs;
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	EXPEDITED-DATA(CONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant ARQ DATA Type 4 D_PDUs;
D_UNIDATA_REQUEST (NON-ARQ)	DATA(CONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	EXPEDITED DATA (CONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs

When in the IDLE(CONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-19. Event/Transition/Action Rules for IDLE (CONNECTED) State: Reception-Related Event

Received D_PDU/ Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	DATA(CONNECTED)	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU.
ACK Type 1 D_PDU	DATA(CONNECTED)	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED
DATA-ACK Type 2 D_PDU	DATA(CONNECTED)	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU as appropriate; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
RESET/WIN RESYNC Type 3 D_PDU	IDLE(CONNECTED), i.e. No Change	COMPOSE and SEND RESET/WIN RESYNC Type 3 D_PDU as required; EXECUTE FULL RESET/WIN RESYNC Protocol if required.
EXPEDITED-DATA Type 4 D_PDU	EXPEDITED-DATA (CONNECTED)	PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND EXPEDITED ACK Type 5 D_PDU
EXPEDITED-ACK Type 5 D_PDU	EXPEDITED-DATA (CONNECTED)	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (CONNECTED)	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	DATA(CONNECTED)	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_ INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	EXPEDITED-DATA (CONNECTED)	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_ UNIDATA_INDICATION if a complete C_PDU is received
WARNING Type 15 D_PDU	IDLE(CONNECTED), i.e. No Change	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.6 DATA(CONNECTED) State Transition Rules

When in the DATA(CONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-20. Event/Transition/Action Rules for DATA (CONNECTED) State: CAS-Related Events

Received D_Primitive Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	DATA(CONNECTED)	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.]
D_CONNECTION_ TERMINATED	IDLE(UNCONNECTED)	TERMINATE ARQ processing; RESET State Variables.
D_UNIDATA_REQUEST (ARQ)	DATA(CONNECTED)	SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant DATA Type 0 D_PDUs;
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	EXPEDITED-DATA(CONNECTED)	SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant EXPEDITED-DATA Type 4 D_PDUs;
D_UNIDATA_REQUEST (NON-ARQ)	DATA(CONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	EXPEDITED DATA (CONNECTED)	SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs

When in the DATA(CONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-21. Event/Transition/Action Rules for DATA (CONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	DATA(CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if a complete C_PDU received; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU.
ACK Type 1 D_PDU	DATA (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; SEND additional DATA Type 0 D_PDUs, if any and as able; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
DATA-ACK Type 2 D_PDU	DATA (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU as appropriate; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
RESET/WIN RESYNC Type 3 D_PDU	If FULL_RESET_CMD, IDLE(CONNECTED), otherwise, DATA(CONNECTED), i.e. No Change	COMPOSE and SEND RESET/WIN RESYNC Type 3 D_PDU as required; EXECUTE FULL RESET/WIN RESYNC Protocol if required.
EXPEDITED-DATA Type 4 D_PDU	EXPEDITED-DATA (CONNECTED)	PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_ INDICATION if received complete C_PDU; COMPOSE and SEND EXPEDITED ACK Type 5 D_PDU
EXPEDITED-ACK Type 5 D_PDU	EXPEDITED-DATA (CONNECTED)	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (CONNECTED	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	DATA(CONNECTED), i.e. No Change	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_ INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	EXPEDITED-DATA (CONNECTED)	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_ UNIDATA_INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	DATA(CONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.7 EXPEDITED-DATA (CONNECTED) State Transition Rules

When in the EXPEDITED-DATA(CONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-22. Event/Transition/Action Rules for EXPEDITED-DATA (CONNECTED) State: CAS-Related Events

Received D_Primitive Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	EXPEDITED-DATA (CONNECTED), i.e. No Change	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.]
D_CONNECTION_ TERMINATED	IDLE(UNCONNECTED)	TERMINATE ARQ processing; RESET State Variables.
D_UNIDATA_REQUEST (ARQ)	in Accordance with Section C.6.1.3: IF completed transmitting all Expedited-Data, change to DATA(CONNECTED), OTHERWISE, EXPEDITED-DATA (CONNECTED), i.e., No Change.	in Accordance with Section C.6.1.3: IF completed sending all Expedited-Data, SEGMENT C_PDU, UPDATE State Variables, then, COMPOSE and SEND all resultant DATA Type 0 D_PDUs; OTHERWISE, QUEUE C_PDU Pending completion of Expedited-Data transmission.
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	in Accordance with Section C.6.1.3: EXPEDITED-DATA (CONNECTED), i.e. No Change	in Accordance with Section C.6.1.3: SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant EXPEDITED-DATA Type 4 D_PDUs;
D_UNIDATA_REQUEST (NON-ARQ)	in Accordance with Section C.6.1.3: IF completed transmitting all Expedited-Data, change to DATA(CONNECTED), OTHERWISE, EXPEDITED-DATA (CONNECTED), i.e., No Change.	in Accordance with Section C.6.1.3: IF completed sending all Expedited-Data, SEGMENT C_PDU, then, COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE, QUEUE C_PDU Pending completion of Expedited-Data transmission.
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	EXPEDITED-DATA (CONNECTED), i.e. No Change.	SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs

When in the EXPEDITED-DATA(CONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-23. Event/Transition/Action Rules for EXPEDITED-DATA (CONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	EXPEDITED-DATA (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU
ACK Type 1 D_PDU	EXPEDITED-DATA (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
DATA-ACK Type 2 D_PDU	EXPEDITED-DATA (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
RESET/WIN RESYNC Type 3 D_PDU	If FULL_RESET_CMD, IDLE(CONNECTED), otherwise, DATA(CONNECTED)	COMPOSE and SEND RESET/WIN RESYNC Type 3 D_PDU as required; EXECUTE FULL RESET/WIN RESYNC Protocol if required.
EXPEDITED-DATA Type 4 D_PDU	EXPEDITED-DATA (CONNECTED), i.e. No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND EXPEDITED ACK Type 5 D_PDU
EXPEDITED-ACK Type 5 D_PDU	EXPEDITED-DATA (CONNECTED), i.e. No Change.	PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; SEND additional EXPEDITED-DATA D_PDUs, if any and as able; as appropriate, NOTIFY CAS using D_EXPEDITED_ UNIDATA_REQUEST_ CONFIRM or D_EXPEDITED_UNIDATA_REQUEST_ REJECTED.
MANAGEMENT Type 6 D_PDU	MANAGEMENT (CONNECTED)	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	EXPEDITED-DATA (CONNECTED), i.e. No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_ INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	EXPEDITED-DATA (CONNECTED), i.e. No Change	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_ UNIDATA_INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	EXPEDITED-DATA (CONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.1.8 MANAGEMENT (CONNECTED) State Transition Rules

When in the MANAGEMENT(CONNECTED) State, the Data Transfer Sublayer **shall** (1) respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

Table C-24. Event/Transition/Action Rules for MANAGEMENT (CONNECTED) State: CAS-Related Events

Received D_Primitive/ Event	State Transition to:	Action & Comments
D_CONNECTION_MADE	MANAGEMENT (CONNECTED) , i.e., No Change.	IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.]
D_CONNECTION_ TERMINATED	IDLE (UNCONNECTED)	TERMINATE ARQ processing; RESET State Variables.
D_UNIDATA_REQUEST (ARQ)	IF Management Protocol completed, DATA(CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED) , i.e., No Change.	IF Management Protocol completed, SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant DATA Type 0 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol.
D_EXPEDITED_UNIDATA_ REQUEST (ARQ)	IF Management Protocol completed, EXPEDITED DATA (CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED), i.e., No Change.	IF Management Protocol completed, SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant EXPEDITED-DATA Type 4 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol.
D_UNIDATA_REQUEST (NON-ARQ)	IF Management Protocol completed, DATA(CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED) , i.e., No Change.	IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol.
D_EXPEDITED_UNIDATA_ REQUEST (NON_ARQ)	IF Management Protocol completed, EXPEDITED DATA (CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED) , i.e., No Change.	IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol.

When in the MANAGEMENT(CONNECTED) State, the Data Transfer Sublayer **shall** (2) respond to reception of a D_PDU from the lower layers as specified in the following Table:

Table C-25. Event/Transition/Action Rules for MANAGEMENT (CONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE ACK Type 1 D_PDU for deferred transmission on transition to an appropriate state.
ACK Type 1 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
DATA-ACK Type 2 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE ACK Type 1 D_PDU for deferred transmission on transition to an appropriate state; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_ CONFIRM or D_UNIDATA_REQUEST_ REJECTED.
RESET/WIN RESYNC Type 3 D_PDU	MANAGEMENT (CONNECTED)	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED; CONTINUE w/ MANAGEMENT Protocol
EXPEDITED-DATA Type 4 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE EXPEDITED ACK Type 5 D_PDU for deferred transmission on transition to an appropriate state.
EXPEDITED-ACK Type 5 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED
MANAGEMENT Type 6 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS Type 6 D_PDU; RESPOND if required.
NON-ARQ DATA Type 7 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU
EXPEDITED NON_ARQ Type 8 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU
WARNING Type 15 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.6.1.2 State Rules for Sending and Receiving D_PDUs

A node **shall** (1) transmit only those D_PDUs which are allowed for its current state as follows:

IDLE(UNCONNECTED)	type 15 D_PDUs
DATA(UNCONNECTED)	type 7, or 15 D_PDUs
EXPEDITED DATA(UNCONNECTED)	type 8, or 15 D_PDUs
MANAGEMENT(UNCONNECTED)	type 6, or 15 D_PDUs
IDLE(CONNECTED)	type 15 D_PDUs
DATA(CONNECTED)	type 0, 1, 2, 3, 7, 8, or 15 D_PDUs
EXPEDITED DATA(CONNECTED)	type 1, 4, 5, 8, or 15 D_PDUs
MANAGEMENT(CONNECTED)	type 6, 8, or 15 D_PDUs

A node **shall** (2) receive and process all valid D_PDUs regardless of its current state. Transmission of responses to a received D_PDU may be immediate or deferred, as appropriate for the current state and as specified in Section 6.1.1.

C.6.1.3 D_PDU Processing Rules: EXPEDITED-DATA (CONNECTED) State

A separate Frame Sequence Number counter **shall** (1) be maintained for the transmission of ARQ Expedited Data, distinct from the counter with the same name used for regular-delivery service with D_PDU types 0 and 2.

A separate C_PDU ID counter **shall** (2) be maintained for the transmission of ARQ Expedited Data, distinct from the counter with the same name used for regular non-ARQ and expedited non-ARQ delivery services with D_PDU types 7 and 8.

Upon entering the EXPEDITED-DATA (CONNECTED) state, the EXPEDITED-DATA D_PDU Frame Sequence Number counter **shall** (3) be set to the value zero (0).

Starting or restarting another ARQ machine (i.e. establishing a link with a new node or re-establishing a link with a previously connected node) **shall** (4) reset the ARQ machine for the EXPEDITED DATA-state.

The processing of D_PDUs containing Expedited Data **shall** (5) proceed according to a *C_PDU-level* stop-and-wait protocol, as follows:

a. No new Expedited-C_PDUs **shall** (6) be accepted for service until the last D_PDU of a prior Expedited-C_PDU has been acknowledged.
b. Each time a D_EXPEDITED_UNIDATA_REQUEST Primitive is accepted for service, the Expedited Data D_PDU Frame Sequence Number counter **shall** (7) be reset to the value zero and the C_PDU ID counter **shall** (8) be incremented modulo-16.

Upon exiting the EXPEDITED DATA(CONNECTED) state to another state, all unsent EXPEDITED-DATA C_PDUs (and portions of C_PDUs) **shall** (9) be discarded.

Similarly at a receiving node, transition from the EXPEDITED DATA(CONNECTED) state to another state **shall** (10) result in the deletion of a partially assembled C_PDU. [Note: The decision to delete partially processed C_PDUs when a transition is made to another data transfer state reflects the primary usage of the expedited data transfer service, i.e. the transfer of short, high priority, upper layer peer-to-peer PDUs that require immediate acknowledgement.]

C.6.2 D_PDU Frame-Sequence Numbers and Flow-Control when in the DATA STATE

Because frame numbers are operated upon using modulo arithmetic, it is convenient to represent the ARQ sliding-window that controls D_PDU flow as a segment of a circular buffer as shown in Figure C-43.

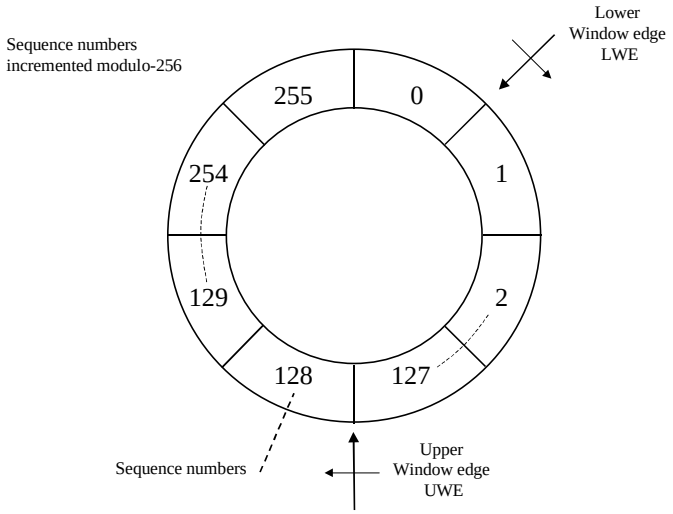


Figure C-43. Sequence-number Space: Integers 0..255

The Data Transfer Sublayer **shall** (1) satisfy the following requirements for D_PDU flow control:

a) Each node **shall** (2) maintain a transmit and a receive flow-control window buffer for each connection supported.

b) The frame sequence numbers **shall** (3) be assigned uniquely and sequentially in an ascending modulo 256 order during the segmentation of the C_PDU into D_PDUs.

c) The frame sequence numbers **shall** (4) not be released for reuse with another D_PDU until the receiving node has acknowledged the D_PDU to which the number is assigned.

d) The transmit lower window edge (TX LWE) **shall** (5) indicate the lowest-numbered outstanding unacknowledged D_PDU (lowest-numbered allowing for the modulo 256 operations).

e) The transmit upper window edge (TX UWE) **shall** (6) be the number of the last new D_PDU that was transmitted (highest D_PDU number, allowing for the modulo 256 operation).

f) The difference (as a modulo-256 arithmetic operator) between the TX UWE and TX LWE – 1 **shall** (7) be equal to the “current transmitter window size”.

g) The “maximum window size” **shall** (8) equal 128.

h) The "maximum allowable window size" may be a node-configurable parameter (this is recommended) and **shall** (9) not exceed the "maximum window size".

i) The "current transmitter window size" at any moment is variable as a function of the current TX UWE and TX LWE and **shall** (10) not exceed the "maximum allowable window size". This allows for no more than 128 ("maximum window size") outstanding unacknowledged D_PDUs in any circumstance.

j) If the "current transmitter window size" equals the "maximum allowable window size", no additional new D_PDUs **shall**(11) be transmitted until the TX LWE has been advanced and the newly computed difference (modulo 256) between the TX UWE and the TX LWE – 1 is less than the maximum allowable window size.

k) The receive lower window edge (RX LWE) **shall** (12) indicate the oldest D_PDU number that has not been received (lowest D_PDU number, allowing for modulo 256 arithmetic operations).

l) The receive lower window edge (RX LWE) **shall** (13) not be decreased when retransmitted D_PDUs are received that are copies of D_PDUs received previously.

m) The receive upper window edge (RX UWE) **shall** (14) be the frame-sequence number of the last new D_PDU received. [Note: More explicitly, the RX UWE is the Frame Sequence Number of the received D_PDU which is the greatest distance (modulo 256) from the RX LWE. The RX UWE increases monotonically as D_PDUs with higher (mod 256) TX Frame Sequence Numbers are received; it does not move back when retransmitted D_PDUs are received.]

n) D_PDUs with TX FSN falling outside the maximum window size of 128 **shall** (14) not be accepted or acknowledged by the receiver node.

o) If the value of the RX LWE field in any ACK Type 1 or DATA-ACK Type 2 D_PDU is greater than the TX LWE, the Data Transfer Sublayer **shall** (16) declare all D_PDUs with Frame Sequence Numbers between the TX LWE and the RX LWE value as acknowledged, and **shall** (17) advance the TX LWE by setting it equal to the value of the RX LWE.

p) The initial condition of the window edge pointers (e.g., on the initialization of a new link) **shall** (18) be as follows:

- TX LWE = 0
- TX UWE = 255
- RX LWE = 0
- RX UWE = 255

[Note that, although the flow control limitations would allow as many as 128 D_PDUs to be transmitted in one transmission, other protocol parameters will also effect this in an indirect way; the structure of the EOT parameter allows a maximum transmission interval of about 2 minutes. If a D_PDU size of 200 bytes is used at 75 bps, only 5 D_PDUs can be transmitted. However, even if a node receives no acknowledgements, it would be possible to transmit many more D_PDUs before transmission would halt due to flow control restrictions.]

The nominal relationships between the ARQ Flow-Control variables specified above are shown in Figure C-44.

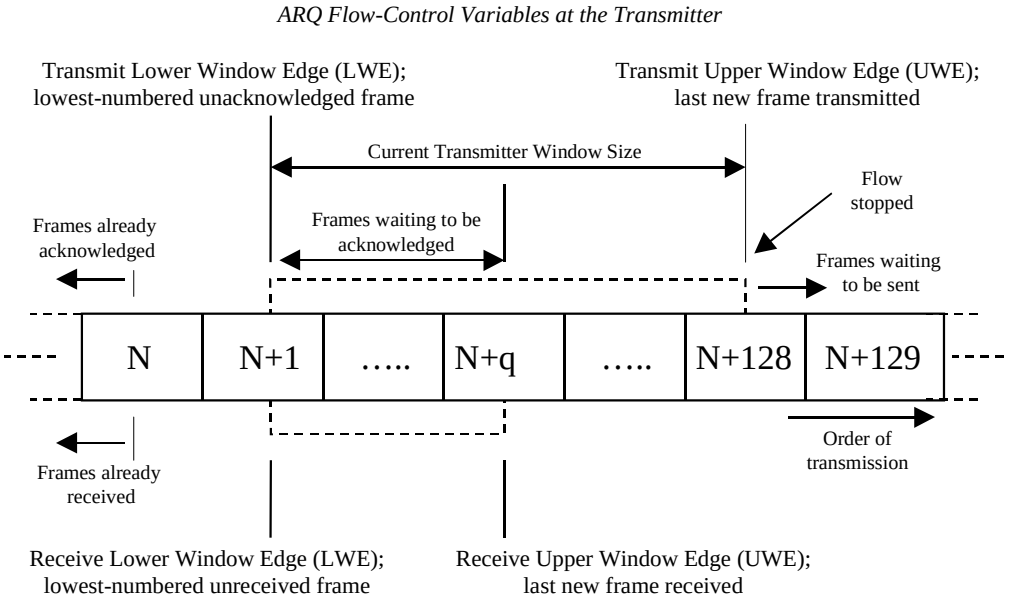


Figure C-44. ARQ Flow Control Window Variables

C.6.3 Synchronisation of the ARQ Machine

Procedures for synchronizing the ARQ machine for a link are specified in the following paragraphs.

C.6.3.1 Initial Synchronisation

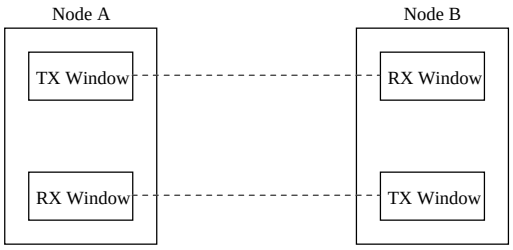
On starting an ARQ machine (i.e. establishing a link with a new node, or establishing a new link with a previously connected node), the transmit and receive ARQ window-edge pointers **shall** (1) be set to the initial values (as defined in C.6.2).

On an HF circuit, it may be desirable to revive a data state connection that has, for example, timed out partway through a file transmission due to a temporary worsening of the propagation conditions. In this case, the loss of data will be minimised if the ARQ machine associated with the connection is re-activated (i.e. the transmit and receive ARQ window-edge pointers are not re-initialised on re-establishing the link). However, this assumption may, for various reasons, *not* be valid (for example, because one of the nodes has experienced a power failure before re-activation) and so a synchronisation verification procedure **shall** (2) be executed whenever a link is re-established.

C.6.3.2 Verification and Maintenance of Synchronisation

The following synchronisation verification procedure **shall** (1) be used to verify on an *ongoing basis* if the peer ARQ processes are in synchronisation and, if required, to effect a reset or re-synchronisation of the peer ARQ window pointers.[3]

[3] Verification of synchronisation on an ongoing basis and, if required, re-synchronisation is the responsibility of the Data Transfer Sublayer. However, under some circumstances a reset or re-



This figure illustrates the fact that, at each end of the node, there is one transmit and one receive window. The dotted lines show which pairs of windows need to remain in synchronisation.

C.6.3.2.1 Synchronisation Tests Performed at the Destination Node

Synchronisation tests performed at the destination node make use of the TX lower window edge (LWE) and TX upper window edge (UWE) flags in conjunction with the TX FRAME SEQ # contained in the header of DATA-ONLY and DATA-ACK frames. The appropriate flag is raised (value = 1) when the TX FRAME SEQ # corresponds to the originating node's transmit ARQ LWE or UWE pointers. The following tests **shall** (2) be used to detect *loss of synchronisation.*

Test 1 : Transmit Upper Window Edge

The purpose of this test is to ensure that the TX UWE of the originating node is within the valid range defined by the destination node window edge pointers. This test **shall** (3) be carried out whenever a D_PDU is received with its TX UWE flag set. If the TX UWE passes this test then the two nodes are *in synchronisation.*

Equation 1 :

IN SYNC = (TX UWE >= RX UWE) **AND** (TX UWE <= MAX WIN SIZE –1 + RX LWE)

If the peer ARQ machines are properly synchronised, the TX UWE cannot be less than the RX UWE (as this would indicate that a D_PDU has been received which has not been transmitted). This condition is expressed by the first part of equation 1. It also cannot exceed the limits defined by the maximum window size of 128 and therefore cannot be greater than the
{RX LWE + MAX WIN SIZE –1} (modulo 256) [4]. This condition is expressed by the second part of equation 1. Equation 1 can therefore be used to establish whether the TX

synchronisation may be initiated by the Management Sublayer, e.g. following the (re)establishment of a link and as part of some link maintenance procedures.

[4] All the synchronisation tests assume a fixed window size equivalent to the maximum window size of 128 frames. Although the STANAG permits the use of a variable transmit window size, this information is not transmitted over the air. The destination node therefore has no knowledge of the window size of the originating node and so cannot take it into account in the synchronisation tests. The tests should still be applied when the window size is varied but in this case some out-of-synchronisation conditions will not be detected.

UWE is *in synchronisation* with the RX window edges and *loss of synchronisation* has occurred if this equation is not satisfied.

The *in synchronisation* and *out of synchronisation* regions of the FSN circle described by equation 1 are illustrated graphically in Figure C-45.

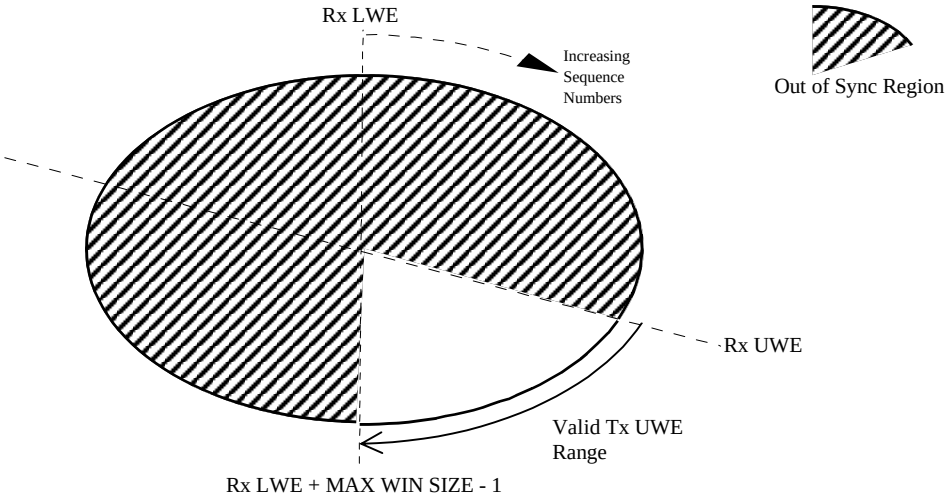


Figure C-45. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 1)

Test 2 : Transmit Lower Window Edge

The purpose of this test is to ensure that the TX LWE of the originating node is within the valid range defined by the destination node window edge pointers. This test **shall** (4) be carried out whenever a D_PDU is received with its TX LWE flag set. If the TX LWE passes this test then the two nodes are *in synchronisation*.

Equation 2 :

IN SYNC = (TX LWE >= RX UWE – (MAX WIN SIZE –1)) **AND** (TX LWE <= RX LWE)

If the peer ARQ machines are properly synchronised, the TX LWE should not be outside the bounds defined by the maximum window size as seen from the perspective of the destination node. The TX LWE indicated by the incoming D_PDU therefore cannot be less than the {RX UWE – (MAX WIN SIZE – 1)} (modulo 256). This condition is expressed by the first part of equation 2. The TX LWE also cannot be greater than the RX LWE (as this would indicate that the originating node has marked as acknowledged a frame that the destination node has not yet received). This condition is expressed by the second part of equation 2. Equation 2 can therefore be used to establish whether the TX LWE is *in synchronisation* with the RX window edges and l*oss of synchronisation* has occurred if this equation is not satisfied.

The *in synchronisation* and *out of synchronisation* regions of the FSN circle described by equation 2 are illustrated graphically in Figure C-46.

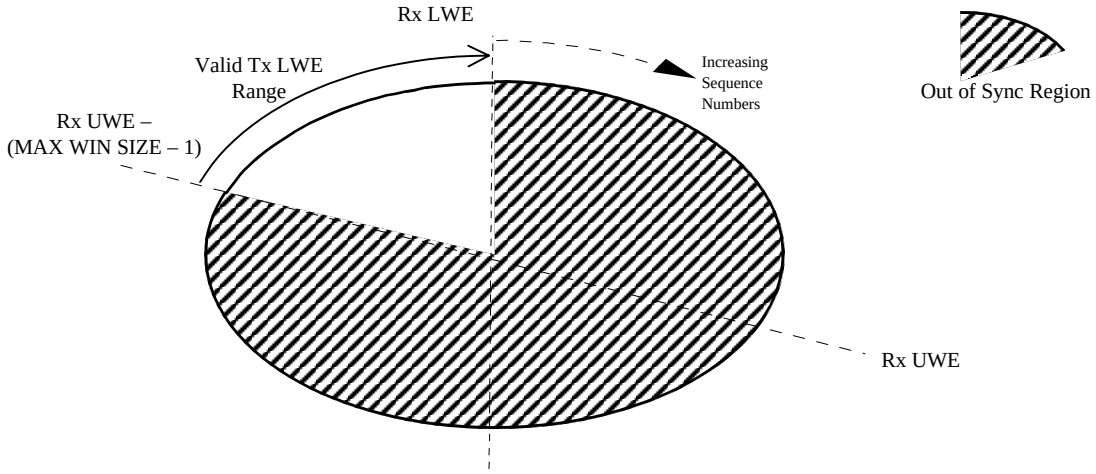


Figure C-46. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 2)

Test 3 : All Received Frames

The purpose of this test is to ensure that the frame sequence number of a frame received from the originating node is within the valid range defined by the destination node window edge pointers. This test **shall** (5) be carried out on every DATA and DATA-ACK frame received. If the frame sequence number of the incoming frame passes this test then the two nodes are *in synchronisation.*

Equation 3 :

IN SYNC = (TX FSN <= RX LWE + (MAX WIN SIZE – 1)) **AND**
(TX FSN >= RX UWE – (MAX WIN SIZE – 1))

If either part of equation 3 is not true, then the frame sequence number falls outside the range defined by the maximum window size of 128 frames and *loss of synchronisation* between the two nodes has occurred.

The *in synchronisation* and *out of synchronisation* regions defined by equation 3 are illustrated graphically in Figure C-47.

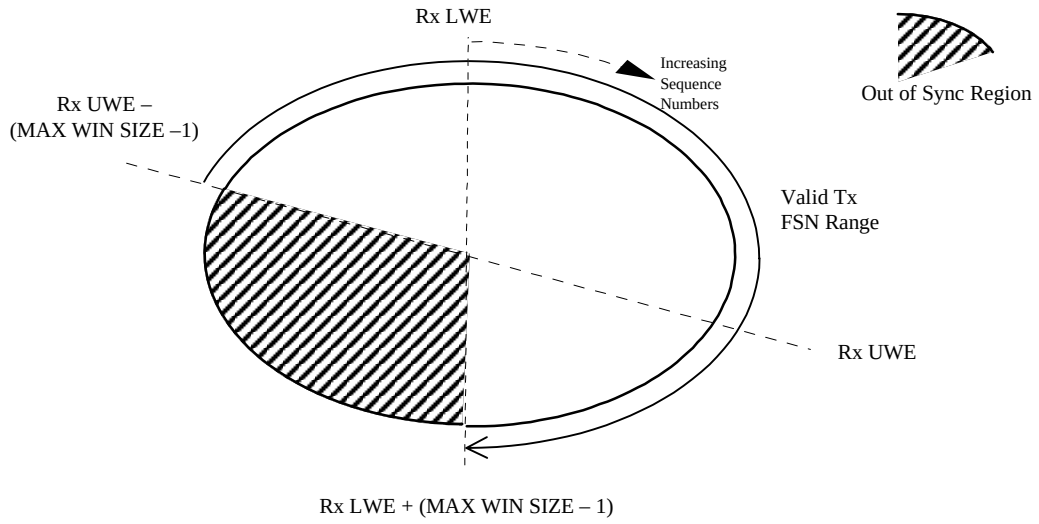


Figure C-47. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 3)

C.6.3.2.2 Synchronisation Tests Performed at the Originating Node

The purpose of tests carried out at the originating node is to ensure that the frame sequence numbers of acknowledged frames are within the range defined by the transmit window edge pointers. These tests **shall** (6) be applied whenever a DATA-ACK or ACK-ONLY frame is received.

Test 4 : Receive Lower Window Edge

The value of the RX LWE is included in all DATA-ACK and ACK-ONLY frames. The following test is used to determine whether the RX LWE is within the range defined by the TX window edge pointers. If the RX LWE passes this test then the two nodes are *in synchronisation.*

Equation 4 :

IN SYNC = (RX LWE >= TX LWE) **AND** (RX LWE <= TX UWE +1)

If equation 4 is not satisfied then *loss of synchronisation* has occurred.

The *in synchronisation* and *out of synchronisation* regions defined by equation 4 are illustrated graphically in Figure C-48.

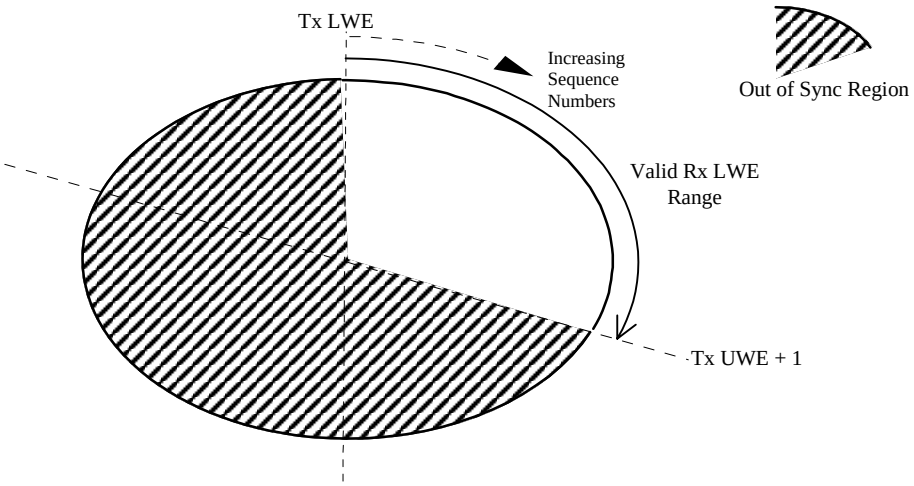


Figure C-48. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 4)

Test 5 : Explicitly Acknowledged Frames

Individual frames may be acknowledged in a DATA-ACK or ACK-ONLY frame by setting bits in the selective ack header field. The frame sequence numbers (FSNs) corresponding to such bits must also fall within the range defined by the transmit window edges if the two nodes are in synchronisation. The following test **shall** (7) be used to determine whether acknowledged FSNs fall within the correct range.

Equation 5 :

IN SYNC = (Acknowledged FSN > TX LWE) **AND** (Acknowledged FSN <= TX UWE)

If acknowledged FSNs do not satisfy the condition defined in equation 5 then *loss of synchronisation* has occurred.

The *in synchronisation* and *out of synchronisation* regions of the frame sequence number circle of the originating node are illustrated graphically in Figure C-49.

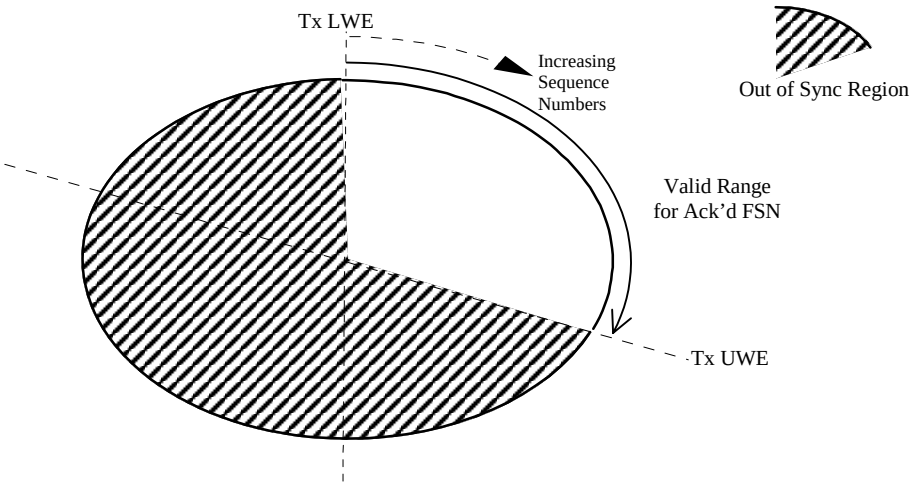


Figure C-49. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 5)

C.6.4 Data Rate Control

Adaptive control of the transmission data-rate by the Data Transfer Sublayer is optional in STANAG 5066. Data Rate Control requires a modem-control interface (virtual or real) between the Data Transfer Sublayer and the attached modem through which changes in the modem data rate and other waveform parameters may be effected by the Data Transfer sublayer. In addition, the Data Transfer Sublayer should be capable of receiving modem-status reports over the modem-control interface. A minimum set of capabilities required and desirable of the modem-control interface are defined in Annex E for information for implementers of this STANAG.

If Data Rate Control is implemented, it **shall** (1) be implemented as specified in the subsections below.

Nodes that do not implement Data Rate Control **shall** (2) use the appropriate EOW and Management message types specified in Section 3.5 and the protocol defined below to signal this condition to other nodes.

C.6.4.1 Initial (Default) Data Rate

All connections on which the data rate or other modem parameters can be controlled **shall** (1) be initiated at 300 bps, using short interleaving. The waveform **shall** (2) be selected by the operator during node initialization.[5]

C.6.4.2 Adaptive Data Rate Change Procedure

The receiving node has available to it complete information on the signal it is receiving and is therefore best able to determine the optimum data rate at which data should be sent to it.

The receiving node may use the EOW Type 1 message to advise the sending node of the optimum data rate, with the "Other parameters" field of the message set in accordance with

[5] If the waveform used supports automatic recognition of coding and interleaving parameters, defined initial settings are not necessary.

Section C.5 to indicate that the message is for advisement, and not a request to change the data rate. The action and the final decision remain with the sending node. Mechanisms other than the Type 1 EOW (i.e., Data Rate Change) message may be used to arrive at a decision to change data rate; however, these other mechanisms are not defined here and will not be interoperable between vendors. Algorithms to determine when or if the data rate change capability would be exercised are beyond the scope of this STANAG. At a minimum, systems implementing STANAG 5066 **shall** (1) implement and support data rate changes in accordance with the procedures defined here.

On receiving an EOW Type 1 Data Rate Change message with the "Other-parameters" field indicating that this is a request for a Data Rate Change, a node **shall** (2) comply with the parameters specified in the message unless some specific reason prevents it doing so. Generally this means that the node will initiate a data rate change (DRC) procedure.
(Note: Reasons for failure to comply with the request include lack of remote modem control, or local conditions do not support a change if modem TX and RX data rate must be the same, and are specified for the EOW/Management messages in Section C.5).

Following a decision to change data rate, a node **shall** (3) use TYPE 6 D_PDUs (i.e., MANAGEMENT D_PDUs) containing Type 1 and Type 2 EOW messages to implement and co-ordinate the change.

The data-rate change procedure **shall** (4) be executed in the MANAGEMENT (CONNECTED) State in accordance with Section 6.1.

If the waveform and modems used support automatic recognition of coding and interleaving parameters[6] by the receiver, the data rate change procedures defined here may not be necessary. Following receipt of an EOW DRC advisory message, a system using such a waveform and modem can simply change the transmit data rate. Annex H provides more details on this subject as information for implementers.

Data rate changes **shall** (5) be effective only for a single connection. If a node has a number of connections active with different nodes, the data rate change decisions and procedures **shall** (6) be executed independently for each connection.

The node initiating the data rate change is referred to as the data rate change master (DRC master) for this DRC procedure. Figure C-50 shows a scenario of a successful DRC procedure. The numbers in brackets in the figure, i.e, message [1], are included for reference to the explanatory text.

[6] Examples of such waveforms include the waveforms defined in Annex G to this document, and the MIL-STD-188-110A single tone waveform.

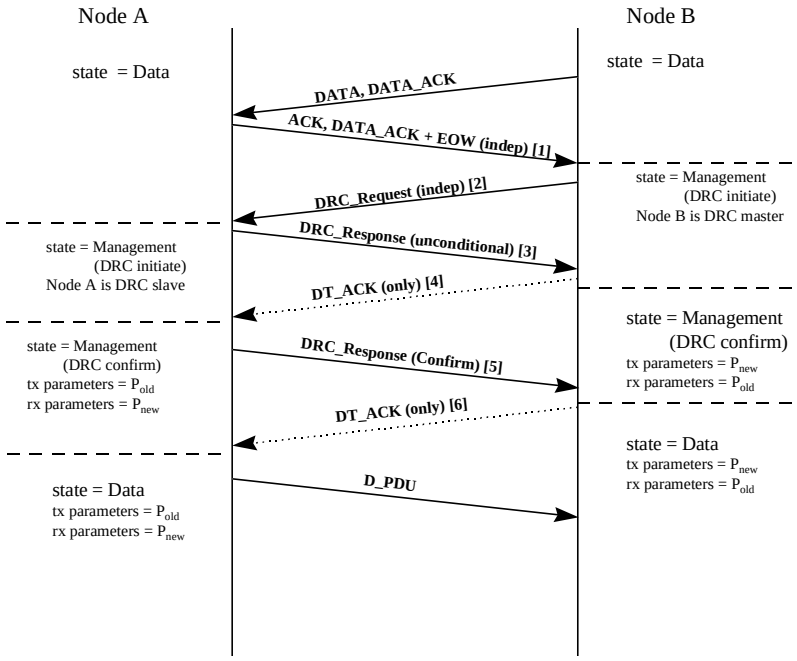


Figure C-50. Data Rate Change Procedure (Scenario 1)

The message numbered [1] in the figure indicates an EOW advisory message. The DRC master **shall** (7) send a DRC Request Message, i.e., a MANAGEMENT (TYPE 6) D_PDU containing a DRC Request (Type 1) EOW Message [2], with the parameters equal to the intended new transmit data rate for the DRC master.

The "Other Parameters" field of the Type 1 Management message **shall** (8) be set to indicate the data rate capabilities of the modem in use at the DRC master for this connection, and to indicate that this is a request for a change, and not an advisory message. In the scenario of Figure C-50, the modem at the DRC master has independent transmit and receive data rate.

The only state transition allowed due to timeout during the Data-Rate-Change procedure **shall**(9) be to the IDLE(UNCONNECTED) state as specified in Section C.6.1, with a D_CONNECTION_LOST primitive sent to the Channel Access Sublayer.
[Note: The management message stop-and-wait protocol, with the repetitions required below, is extremely robust. If it is not possible to complete the procedure in the management state, it is unlikely that returning to another connected state will be productive.]

Timeouts **shall** (10) be set to allow a number of retransmissions before failure is declared and the D_CONNECTION_LOST primitive issued; the number of retransmissions before a timeout **shall** (11) be configurable in the implementation with a default value of 3.

If this procedure was initiated in response to a DRC Request (Type 1) Advisory message, the modem data rate and interleaving parameters **shall** (12) be identical to the parameters in the EOW Type 1 DRC Advisory message, unless they specify a speed for which the DRC master is not equipped.

D_PDUs containing a DRC Request (Type 1) EOW message **shall** (13) be repeated depending on the data rate at which it is transmitted, in accordance with the specification in Table C-26:

Table C-26. Minimum number of DRC messages to be transmitted at various data rates using STANAG 4285 modem (see Implementation Note, below table)

Data rate	Repetitions w/ short interleave	Repetitions w/ long interleave
75	1	9
150	1	18
300	1	37
600	3	75
1200	7	150
2400	15	300

[Implementation Note: The number of transmissions in the table has been specified to (nearly) fill the interleave buffer for the waveform specified by STANAG 4285. For waveforms and interleaver settings not shown, the number of transmissions **shall** (14) be adjusted as required to minimise the use of "stuff bits" to fill the modem interleave buffer. If a waveform with different interleaver or code-block size is used with the STANAG 5066 Subnetwork, a different number of repetitions **may** be used based on optimum packing of the waveform's coding and interleaver frame.]

Repeated MANAGEMENT D_PDUs containing a DRC Request (Type 1) Management message **shall**(15) have the same FRAME ID NUMBER in each of the copies.

Valid EOT information **shall** (16) be supplied in each repeated D_PDU containing a DRC Request (Type 1) Management message, updated as required to denote the end of transmission of all D_PDU messages in the transmission interval (and not the end of the individual D_PDU containing the EOT field).

The same EOT value **shall** (17) not be transmitted in each repeated D_PDU containing a DRC Request (Type 1) Management message unless necessary due to EOT resolution and round-up errors, i.e., because the D_PDU duration is less than half the EOT resolution.

A node (referred to as the DRC slave) that receives a MANAGEMENT D_PDU addressed to it and containing a DRC_Request Type 1 Management message **shall** (18) transition to the Management State as specified in Section C.6.1.

The DRC slave **shall** (19) respond to the DRC_Request D_PDU with a DRC_Response message, i.e., a TYPE 6 MANAGEMENT D_PDU containing a Type 2 DRC Response EOW message, as follows:

- The DRC Response message **shall** (20) indicate either "accept" or "refuse", in accordance with the MANAGEMENT message specifications of Section 3.5, indicating the DRC Slave's capability to change the modem parameters or not.

- If the DRC slave accepts the DRC_Request, the "reason" field **shall** (21) indicate either "unconditional acceptance" or "Tx and Rx parameters must be the same".

- If the DRC slave refuses the request, the reason field **shall** (22) indicate the reason for the refusal.
  [Note: Only the reasons specified in the Section C.5.2 for the DRC Response message are valid reasons for refusing a DRC_Request. The reasons that apply to modem capabilities imply that all attempts to change that modem parameter should stop (for the duration of the current link with the node that gives this reason for refusing a DRC).

- The "Not consistent with local conditions" parameter **shall** (23) only be used to refuse a DRC_Request which indicates a less robust mode (i.e., higher data rate or shorter interleaver) that cannot be supported with the node's current local conditions for noise or error rate.
  [Note: This response should only be used in the event that conditions have changed at the DRC slave between the time that the last EOW advisory message was sent and the DRC_Request was received. Subsequent DRC procedures may be initiated after additional information on channel conditions is obtained and exchanged.]

[Note: The DRC Slave's response is shown as DRC_Response [3] in the figure. The figure shows an example in which the modem at the DRC slave also has independent transmit and receive data rate. Multiple copies of this D_PDU would be transmitted depending on the modem parameters in accordance with the specification; these repeated copies are not shown in the figure. Table C-17 gives a recommended minimum number of times to transmit the message; more may be advisable and transmitted under certain circumstances.]

After receiving the DRC_Response message the DRC master **shall** (24) review its contents and determine the appropriate response, e.g., the DT_ACK(only) [4], in accordance with Table C-27, below.

Table C-27. DRC_Responses and Allowed DRC Master Actions

DRC_Response	*DRC_Response Reason*	*Action allowed by DRC master*
accept	unconditional	a) Send DT_ACK [DT_ACK = TYPE 6 D_PDU w/ ACK field set]
accept	transmit and receive parameters must be the same	a) Send DT_ACK , or b) Send DRC_Response (cancel), or c) Send new DRC Request note 1
refuse	not possible to change modem data rate	a) send DRC_Response (cancel)note 2, or b) send DRC_Request note 3 (with DT_ACK)
refuse	not possible to change modem interleave	a) Send DRC_Response (cancel)note 2, or b) Send DRC_Request note 4 (with DT_ACK)
refuse	not possible to change modem data rate or interleave	a) Send DRC_Response (cancel)note 2 (with DT_ACK)
refuse	not consistent with local conditions (note 5)	a) Send DRC_Response (cancel)note 2, or b) Send DRC_Request note 6 (with DT_ACK)

Notes to Table C-27 and further DRC requirements are as follows:

Note 1: If the procedure is initiated in response to EOW Type 1 message, the DRC master should already know that the DRC slave's transmit and receive parameters must be the same. Therefore, the DRC master **shall** (25) reply with a DT_ACK, i.e., a MANAGEMENT (TYPE 6) D_PDU with the ACK field set equal to one, accepting that the new parameters will apply to both transmit and receive.

Note 2: A DRC Slave that refused a change request **shall** (26) acknowledge the DRC Response (cancel) message with a DT_ACK only and then terminate the DRC procedure.

Note 3: If the DRC Slave refused the change request because it is not possible to change its modem data rate, a new DRC_Request may be sent by the DRC Master to request a different interleave setting at the same data rate.

Note 4: If the DRC Slave refused the change request because it is not possible to change its interleave setting, a new DRC_Request may be sent by the DRC Master to request a different data rate setting at the same interleave setting.

Note 5: As required by previous paragraphs, the DRC Slave may refuse the request with "Reason = not consistent with local conditions " if an only if it is in response to a request for a less robust set of parameters. For example, a request for a higher data rate and/or shorter interleaver than currently in use would be rejected by the DRC Slave if it has determined that these cannot be supported by the current link conditions at the receiver.

Note 6: If the nodes make use of the EOW Type 1 (Advisory) message to initiate the DRC procedure, the master **shall** (27) send the DRC_Response (cancel), and await an updated EOW recommendation before initiating another DRC procedure. If EOW Type 1 messages are not used, DRC_Request may be sent by master to request different modem parameters which may be consistent with the local conditions.

In the table and notes in the preceding paragraphs, a DT_ACK refers to a Data Transfer Sublayer acknowledgement of the preceding MANAGEMENT message (shown at [4] in Figure C-50). A DT_ACK reply indicates that the node has nothing further to communicate with respect to the DRC procedure.

If a DT_ACK (with no further management message) is sent in reply to a DRC_Response "accept" (as shown in Figure C-50), the nodes **shall** (28) change their respective modem parameters and proceed to the "confirmation" phase.

The DRC slave **shall** (29) NOT change its modem parameters until it has received a DT_ACK (with no further management message) from the DRC master.

If a DT_ACK (with no further management message) is sent by the DRC slave in reply to a DRC_Response "cancel", both nodes **shall** (30) abandon the procedure and return to the prior state without changing modem parameters.

After abandoning a DRC procedure because of failure, if node A (formerly the DRC slave) has no queued data or acknowledgements to send to node B, it **shall** (31) send a data D_PDU, expedited data D_PDU, or non-ARQ D_PDU, with zero data attached.

[Note: In the scenario given in Figure C-50, the slave's DRC_Response with an "accept/unconditional" message generates the allowed DT_ACK from the DRC master. After sending the DT_ACK [4], the master changes its modem parameters and waits to receive a DRC Confirm message (type 2 MANAGEMENT message with response set to "confirm" and reason set to "none") from node A ("confirmation phase"). After receiving the DT_ACK [4], the slave changes its modem parameters and transmits a DRC Confirm message [5] to the master.]

On receiving the DRC Confirm message, the DRC Master **shall** (32) respond with a DT_ACK and then return to the processing state it was in before executing the DRC procedure.

After sending the DRC Confirm message [5] to the master and receiving the DT_ACK from the master, the slave **shall** (33) return to the processing state it was in before executing the DRC procedure and send any queued D_PDUs to node B. If node A (formerly the DRC slave) has no queued data to send to node B, it **shall** (34) send a DATA D_PDU or EXPEDITED DATA D_PDU with zero data attached, and the C_PDU Segment Size field set equal to zero (0).

C.6.4.3 Additional DRC Scenarios

The following figures provide additional scenarios for the DRC procedure in different conditions, and are provided as implementation guidance only.

Figure C-51 gives a scenario in which node B, due to HF modem limitations, must operate with the same HF modem transmit and receive parameters. The following numbered paragraphs correspond to the numbers in [brackets] in Figure C-51.

1. In this transmission, the advisory message sent by A carries the recommended Tx speed for B (You should Tx at x bps) and also tells B about the capabilities of A's modem (that A's modem can operate at independent Tx and Rx data rates.
2. The DRC_Request message from B carries the requested new modem parameters for B: "I (B) will transmit at x bps with y interleaving"; these parameters will be as recommended by A, in accordance with Section C.6.4.2. The message also tells A about the capabilities of B's modem, in this case, if I change my Tx parameters (and your Rx parameters), I will have to change my Rx parameters (and your Tx parameters) as well.)
3. DRC_Response message accepts the requested change.
4. The DT_ACK confirms that B received the prior message and triggers B to change its parameters.
5. This DRC_Response (confirm) is a confirmation that the link is up at the new parameters.

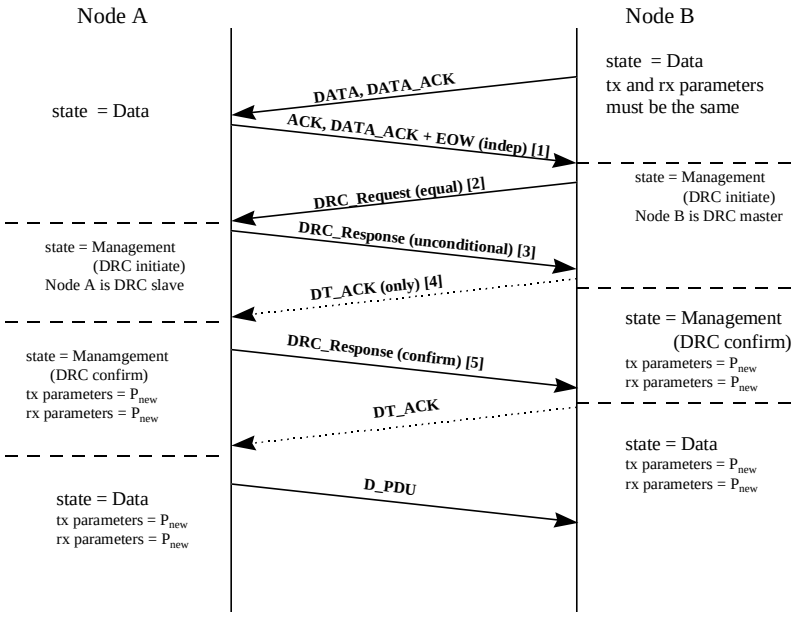


Figure C-51. Data Rate Change Procedure (Scenario 2)

Figure C-52 shows a scenario in which the procedure is refused by the slave, and hence cancelled by the DRC master. In this example, node B has a modem which must have the same transmit and receive parameters. The requested new parameters would then apply to the transmit and receive parameters at both nodes. The following numbered paragraphs correspond to the numbers in [brackets] in Figure C-50.

1. In this transmission, the EOW Type 1 advisory message sent by A carries the recommended Tx speed for B (You should Tx at x bps) and also tells B about the capabilities of A's modem (that A's modem can operate at independent Tx and Rx data rates.
2. The DRC_Request message from B carries the requested new modem parameters for B: "I (B) will transmit at x bps with y interleaving"; these parameters will be as recommended by A in accordance with Section C.6.4.2. The message also tells A about the capabilities of B's modem, in this case, if I change my Tx parameters (and your Rx parameters), I will have to change my Rx parameters (and your Tx parameters) as well. The net result is that both nodes have the same Rx and Tx parameters.
3. DRC_Response message refuses the requested change; a likely reason would be that the local conditions at node A do not support use of the proposed parameters.
4. The DRC master responds by cancelling the procedure.
5. The DT_ACK confirms that B received the prior message and terminates the procedure.

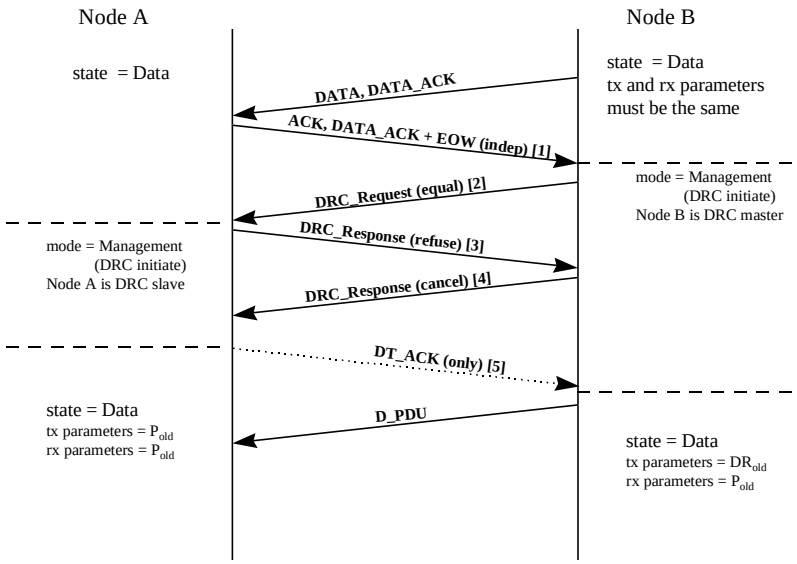


Figure C-52. Data Rate Change Procedure (Scenario 3)

C.6.5 Procedure for use of Type 3 RESET/WIN RESYNC D_PDU

The Type 3 RESET/WIN RESYNC D_PDU, specified in Section C.3.6, supports a number of different resynchronization functions. The procedure for the FULL RESET function is specified in this section.

A FULL RESET procedure **shall** (1) initiated by a node sending a Type 3 RESET/WIN RESYNC D_PDU with field values as follows:

- The FULL RESET CMD flag **shall** (2) be set to 1;
- The RESET FRAME ID NUMBER **shall** (3) be selected from RESET FRAME ID NUMBER sequence;
- The NEW RECEIVE LWE field **shall** (3) be reset to zero;
- The RESET TX WIN RQST flag **shall** (4) be reset to zero;
- The RESET RX WIN CMD flag **shall** (5) be reset to zero.

The Type 3 D_PDU described immediately above is defined to be a FULL RESET CMD D_PDU. A node receiving a FULL RESET CMD D_PDU **shall** (6) proceed as follows:

- The node **shall** (7) set the transmit and receive window pointers to initial values (as defined in Section C.6.2);
- The node **shall** (8) discard from its transmit queue any partially completed C_PDUs;
- The node **shall** (9) flush its receive buffers;
- The node **shall** (10) respond to the originator of the FULL-RESET-CMD D_PDU by sending it a RESET/WIN RESYNC (Type 3) D_PDU with field values set as follows:

    - The RESET ACK flag **shall** (11) be set to 1;
    - The NEW RECEIVE LWE and RESET FRAME ID NUMBER fields **shall**(12) be reset to zero;
    - The RESET TX WIN RQST, FULL RESET CMD and RESET RX WIN CMD flags **shall** (13) be reset to zero.

The D_PDU described immediately above is defined to be a FULL-RESET-ACK D_PDU. The FULL RESET ACK D_PDU **shall** (14) be sent only in response to the FULL RESET CMD D_PDU.

On receiving the FULL-RESET-ACK D_PDU, the node initiating the FULL RESET procedure **shall** (15) proceed as follows:

- The node **shall** (16) set the transmit and receive window pointers to initial values (as defined in C.6.2)
- The node **shall** (17) discard from its transmit queue any partially completed C_PDUs
- The node **shall** (18) flush its receive buffers

This concludes the FULL RESET procedure.