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

C.1.1. Changes in This Edition

The functional differences between this specification and Edition 3 are set out in Section C.10.

The following key changes have been made:

1. Provision of Full Duplex transfer is fully specified. This is used in conjunction with the DUPLEX FIXED and DUPLEX WITH ALE optional profile elements.
2. Support for interoperability with Edition 3 systems is provided and specified in Section C.3.
3. Support for larger window size, critical for Wideband HF, is provided by D_PDU variants with an extended frame sequence number.
4. Some new EOWs are defined to support data rate selection, which is necessary for Wideband HF.
5. A number of additional useful D_PDUs are specified.
6. Clarifications of function and mapping to lower layers are specified.
7. High Data Rate Change Request PDU is now deprecated and specified in Section C.9.1.
8. Expedited data (user service). This service is now deprecated and described in Section C.9.2.

C.1.2. Overview

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 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.1.3. **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, except as noted below in support of delivering partial C_PDUs.

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 and C_PDUs **shall** be delivered in the order in which they arrive. "in-order" delivery is a service for ARQ only. There is some implication in Edition 3 that this service is available for non-ARQ. Experience has shown that attempting to provide this service leads to operational problems.

Delivery of complete and error free C_PDUs is not guaranteed. Delivery of C_PDUs is not guaranteed. A special mode of the non-ARQ service **may** be available to reconstruct partial C_PDUs from D_PDUs in error and deliver the partial and potentially erroneous elements to the Channel Access Sublayer. This service is specified by the NON-ARQ WITH ERRORS optional profile element.

C.1.4. **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.

Transfer of complete and error free C_PDUs is guaranteed.

In the **Selective ARQ** service, the following sub-modes may be specified:

- regular data service.
- expedited data service.

Both regular and expedited services offer delivery confirmation for data delivery to the peer node.

Both regular and expedited services offer an “in-order” option, which will deliver C_PDUs to the remote peer and associated channel access service in the order they were submitted. If this service is not selected, C_PDUs will be delivered as they arrive. Note that expedited services are always “in order”.

NOTE: “in order” is implemented at DTS level and so will apply to all SAPs using it. This can lead to one SAP blocking another. It is **recommended** to avoid use of “in order” whenever possible.

C.1.5. **Queueing and Flow Control**

It is expected that the DTS will queue C_PDUs in order to effectively manage transmission, but that the amount of data queued will be limited. The DTS will provide flow control information to the CAS, which will be used by the SIS to limit data being provided to the DTS.

Experience with DTS suggests that DTS queue lengths should be kept as short as possible, consistent with efficient operation of the DTS. Use of SIS flow control allows better application level monitoring and better behavior in error situations.

C.1.6. **Full Duplex Operation**

Annex C can operate with two types of configuration:

1. Single Channel. Here D_PDUs are all sent to a channel which is shared by multiple nodes with access to the channel controlled by the MAC layer specified in Annex J. D_PDUs may be sent to multiple nodes, optionally using half duplex communication with each of the peer nodes.

2. Full Duplex, using two channels (Receive and Transmit) to communicate with a single peer. In Full Duplex mode, the CAS **shall** ensure that the DTS is used only for communications with a single peer.

The core operations and state machine operation of the DTS are common to both single channel and full duplex. The details of transmission are different, and the rules for both types of operation are set out in Section C.7.5.

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.

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 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 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 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 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. Support for Edition 3 Interoperability

Edition 4 of STANAG 5066 introduces a number of new protocol elements to improve the DTS service. It is a key goal of Edition 4 to ensure that there is robust interoperability with implementations following STANAG 5066 Edition 3 and earlier.

To achieve this an implementation **shall** determine if a peer implementation supports Edition 4 (or subsequent). If a peer only supports edition 3 or the status cannot be determined, an implementation **shall not** use any of the Edition 4 capabilities set out in this annex. This will ensure interoperability with Edition 3.

Peer Capability can be determined in three ways.

1. A priori knowledge. The edition supported by a specific peer or by the whole network may be known.

2. Use of CAS-1 capability negotiation, as specified in the CAS-1 sub-layer. This is always used for ARQ data when ALE is not used. It **may** be used for ARQ data with ALE and **shall** be used if the peer capability is not known. This means that peer capability will always be known if ARQ data is being transferred.

3. The third mechanism is use of the Capability EOW defined in Section C.6.4. This enables Edition 4 peer capability to be explicitly determined with no ARQ data is being exchanged. It also facilitates capability update when a node is upgraded to Edition 4.

There are three types of Edition 4 capabilities that need to be considered:

1. Procedural options, which enhance performance, but **shall** only be used with Edition 4 (or subsequent) peers. These are all limited in scope and clearly flagged in relevant sections.

2. D_PDU Types summarized and categorized in Section C.4. There are three types of D_PDU.

    a. D_PDUs for use with all Editions.

    b. D_PDUs that **shall** only be used with Edition 4 and subsequent Editions.

    c. D_PDUs for use with Edition 3, that **may** be used in Edition 4 when both peers choose to.

3. EOWs are summarized and categorized in Section **C.6**. There are three types of EOW:

    a. EOWs for use with all Editions.

    b. EOWs for use with Edition 3 only.

    c. EOWs for use with Edition 4 and subsequent Editions only.

Edition 4 capabilities **shall** only be used with peers known to support Edition 4. This ensures interoperability with Edition 3 systems.

C.4. 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	Edition
ED3-DATA-ONLY	0	*ARQ data transfer*	SRQ	I	Ed3
ED3-ACK-ONLY	1	*Acknowledgement of ARQ data transfer*	SRQ	C	Ed3
ED3-DATA-ACK	2	*ARQ data transfer with acknowledgement*	SRQ	I+C	Ed3
ED3-RESET/WIN-RESYNC	3	*Reset/Re-synchronized peer protocol entities*	IRQ	C	Ed3
EXPEDITED-DATA-ONLY	4	*Expedited ARQ data transfer*	SRQ	I	All
EXPEDITED-ACK-ONLY	5	*Acknowledgement of expedited ARQ data transfer*	SRQ	C	All
MANAGEMENT	6	*Management message*	IRQ	C	All
NON-ARQ-DATA	7	*Non-ARQ data transfer*	NRQ	I	All
EXPEDITED-NON-ARQ-DATA	8	*Expedited non-ARQ data transfer*	NRQ	I	All
DATA-ONLY	9	*ARQ data transfer*	SRQ	I	Ed4
ACK-ONLY	10	*Acknowledgement of data transfer*	SRQ	C	Ed4
DATA-ACK	11	*ARQ data transfer with acknowledgement*	SRQ	I+C	Ed4
RESET/WIN-RESYNC	12	*Reset/Re-synchronized peer protocol entities*	IRQ	C	Ed4
EXTENSION	13	*Extension message*	-	-	Ed4
PADDING	14	*Header only*	-	C	Ed4
WARNING	15	*Unexpected or unrecognized D_PDU type*	-	C	All

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

Edition Support has the following options controlling STANAG 5066 Editions:

1. All: For use in all Editions; or
2. Ed4: For use in Edition 4 and subsequent editions only; or
3. Ed3: Primarily for use for Edition 3 interoperability but **may** be used between Edition 4 peers by mutual negotiated agreement.

All D_PDU types may be used to support single channel and full duplex transmission modes. 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; or
2. I (Information) Frames; or
3. 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:

1. NRQ: No Repeat-Request.(i.e., Non-ARQ) Protocol; or
2. SRQ: Selective Repeat-Request Protocol; or
3. IRQ: Idle Repeat-Request Protocol

After sending I-frames, the NRQ protocol 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. Multiple transmissions of C frames is often desirable, as loss of acks can lead to unnecessary duplicate data transmission.

After sending I-frames, the Selective RQ protocol 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, as specified in Section C.7.5.6.

After sending an I-frame, the Idle RQ protocol, also known as a stop and wait protocol 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 configurable 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.7.1

C.4.1. **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).

SYNC (16 bits)	HEADER (max 38 bytes)	HEADER CRC (16 bits)

Figure C-1 (a). Format for D_PDU C-Frame types (1, 3, 5, 6, 10, 12, 13, 14 and 15)

SYNC (16 bits)	HEADER (max 38 bytes)	HEADER CRC (16 bits)	SEGMENTED N_PDU (max 1023 bytes)	CRC (32 bits)

Figure C-1 (b). Format for D_PDU I and I+C Frame types (0, 2, 4, 7, 8, 9, 10, 11 and 13)

C.4.2. **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, 12, 14 and 15and 15 **shall** (2) use only the C-Frame structure defined in Figure C-2 (a). D_PDU types 10 and 13 **may** use this structure.

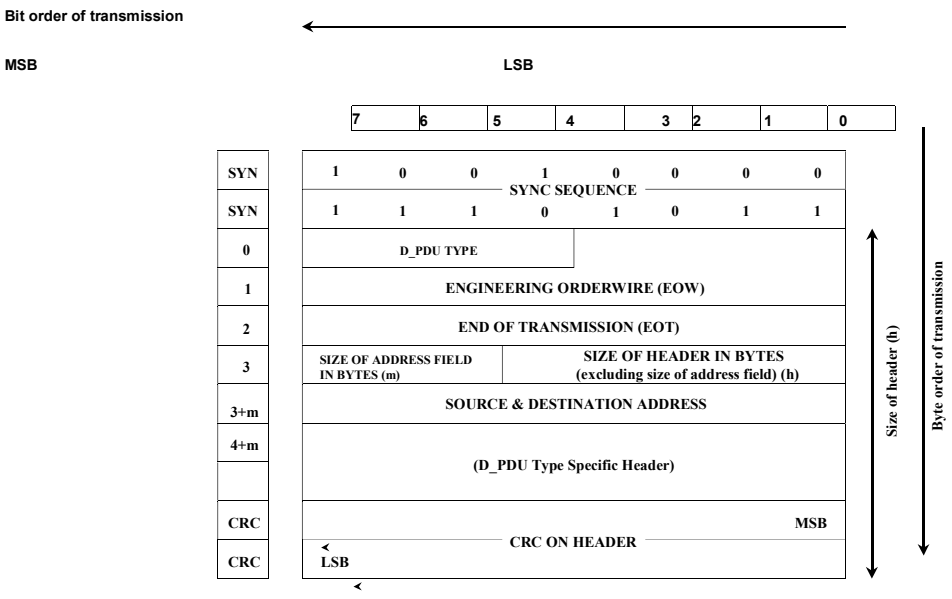


Figure C-2 (a). Generic D_PDU C-Frame Structure.

The D_PDU types 0, 2, 4, 7, 8, 9, 11 **shall** (3) use the generic D_PDU I and I+C Frame structure defined in Figure C-2 (b). D_PDU types 10 and 13 **may** use this structure.

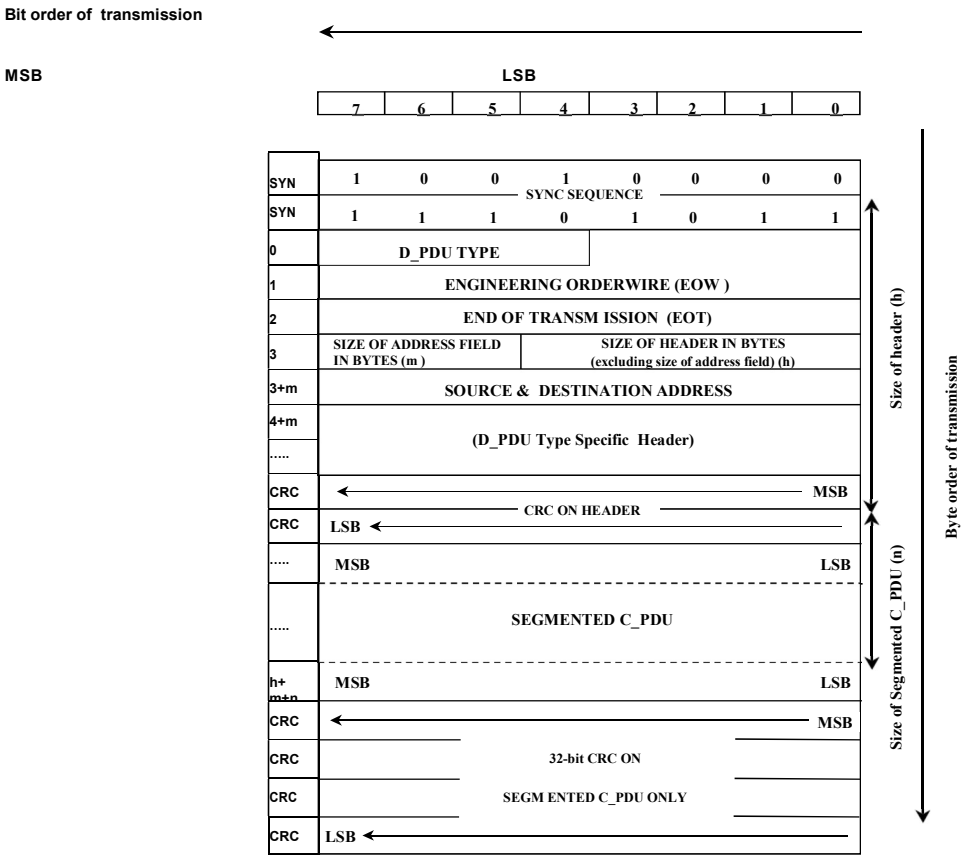


Figure C-2 (b). Generic D_PDU I and I+C Frame Structure.

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.4.2.8.

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

C.4.2.1. 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. All the possible values are assigned in this edition of STANAG 5066.

C.4.2.2. 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.6. Figure C-3 (b) shows how the EOW message is mapped into the generic D_PDU header.

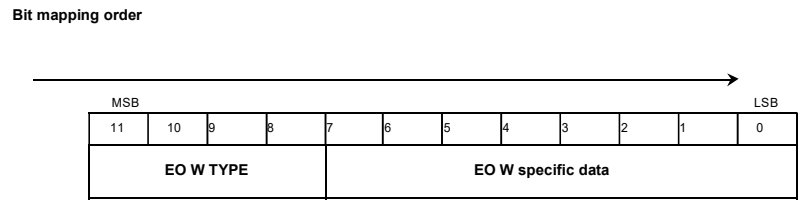


Figure C-3 (a). Generic EOW message.

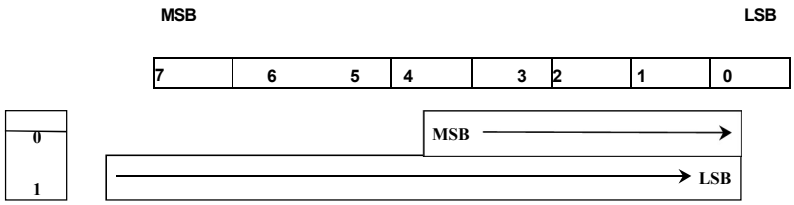


Figure C-3 (b). EOW mapping convention in D_PDU header.

C.4.2.3. 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.

In some modes of transmission, as described in Section C.7.5, the EOT is set to zero. When the EOT is set to zero, no information is provided as to when the transmission ends.

C.4.2.4. 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.4.2.5. 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:

C.4.2.5.1.1. D_PDU Type
C.4.2.5.1.2. EOW
C.4.2.5.1.3. EOT
C.4.2.5.1.4. Size of Address field
C.4.2.5.1.5. Size of Header field
C.4.2.5.1.6. D_PDU-Type-specific header
C.4.2.5.1.7. CRC field

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

C.4.2.6. Source and 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).

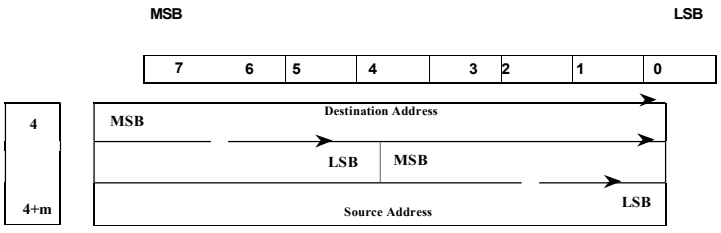


Figure C-4 (a). Address mapping convention in D_PDU header, assuming address-field size is odd

.

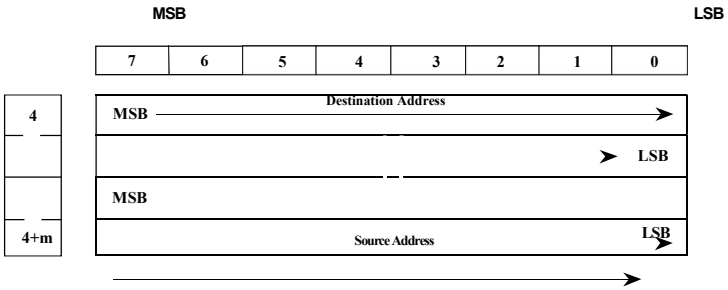


Figure C-4(b). Address mapping convention in D_PDU header, assuming address-field size is even.

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.

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. Consequently, the decimal representation for node addresses (i.e., unicast addresses) lies in the range
[0.0.0.0 ... 15.255.255.255] (this **may** be referred to as 'dotted-decimal address format').

For DPDUs that support group addressing (e.g; THE Type 7/8 Non-ARQ DPDUs), the group-address flag **should** be considered the 29th-bit in the binary representation of the address. Consequently, the decimal representation for group addresses (i.e., multicast addresses) lies in the range [16.0.0.0 ... 31.255.255.255].

The broadcast (all-stations) address in “dotted-decimal” address format is the group address 31.255.255.255.

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.4.2.7. 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.4.4 through C.4.15.

C.4.2.8. Cyclic Redundancy Check (CRC)

A two byte header-only CRC is used. Because the D_PDU header is generally shorter than the data, errors are more likely in the data part of a D_PDU than the header. Protecting the header with its own CRC allows the possibility to detect and use uncorrupted header information even if the data part of a D_PDU contains errors. For this reason, the added overhead of the CRC-on-header field was deemed warranted in the design of STANAG 5066.

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 accommodated 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.

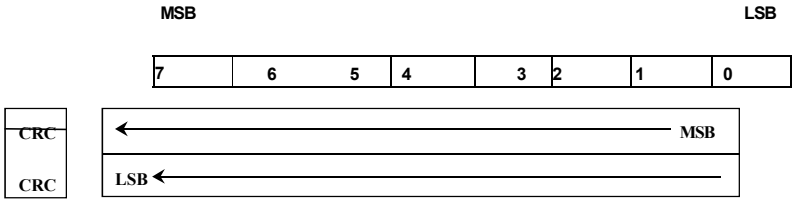


Figure C-5. CRC mapping convention in D_PDU header.

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.4.2.9. **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.

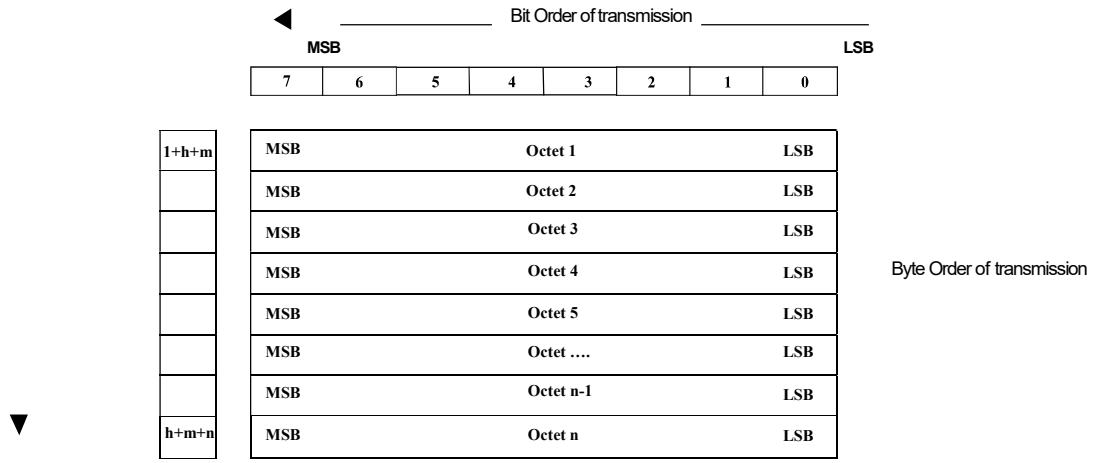


Figure C-6. Segmented C_PDU mapping convention in D_PDU structure.

C.4.2.10. **Size of Segmented C_PDU**

The SIZE OF SEGMENTED C_PDU field **shall** (1) be used only with DATA 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.

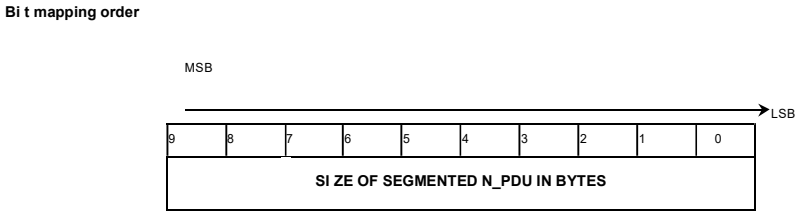


Figure C-7. SIZE OF SEGMENTED C_PDU Field.

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.

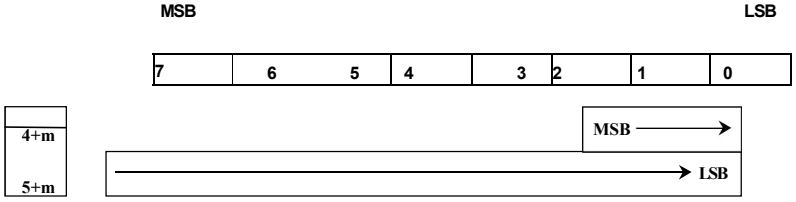


Figure C-8. Segmented C_PDU Size mapping convention in all applicable D_PDU header structures.

C.4.2.11. **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 note 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).

NOTE: 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.

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

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.4.3. CHOICE OF FRAME SEQUENCE NUMBER LENGTH FOR ARQ D_PDUs

STANAG 5066 Ed3 used an eight bit frame sequence number for ARQ data. This choice leads to window exhaustion and significant performance impact at wideband HF speeds and at faster narrowband speeds. Edition 4 introduces a set of four PDUs to support regular ARQ data with a 16 bit frame sequence number, which addresses the performance issue. For three of the PDUs, the longer frame sequence number is the only change. ACK-ONLY has some additional changes to optimize performance.

These PDUs are described in pairs in Sections C.4.4 - C.4.7. The Edition 3 compatible PDUs are prefixed with “ED3-”. These “ED3-” PDUs **shall** be used for regular ARQ data exchange with Edition 3 and earlier peers.

It is anticipated that the new D_PDUs will usually be used for regular ARQ communication between Edition 4 and subsequent peers and this choice is the default. Use of the Edition 3 D_PDUs **may** be negotiated using CAS-1 link setup with agreement of both peers. This has potential to gain about 0.5% throughput improvements in links which will only operate at the slowest narrowband HF speeds.

It is anticipated that Edition 5 will make support for these Edition 3 D_PDUs optional and it is possible that support will be removed in future editions.

C.4.4. DATA-ONLY D_PDUs (Simplex data transfer)

C.4.4.1. ED3-DATA-ONLY (Type 0) D_PDU

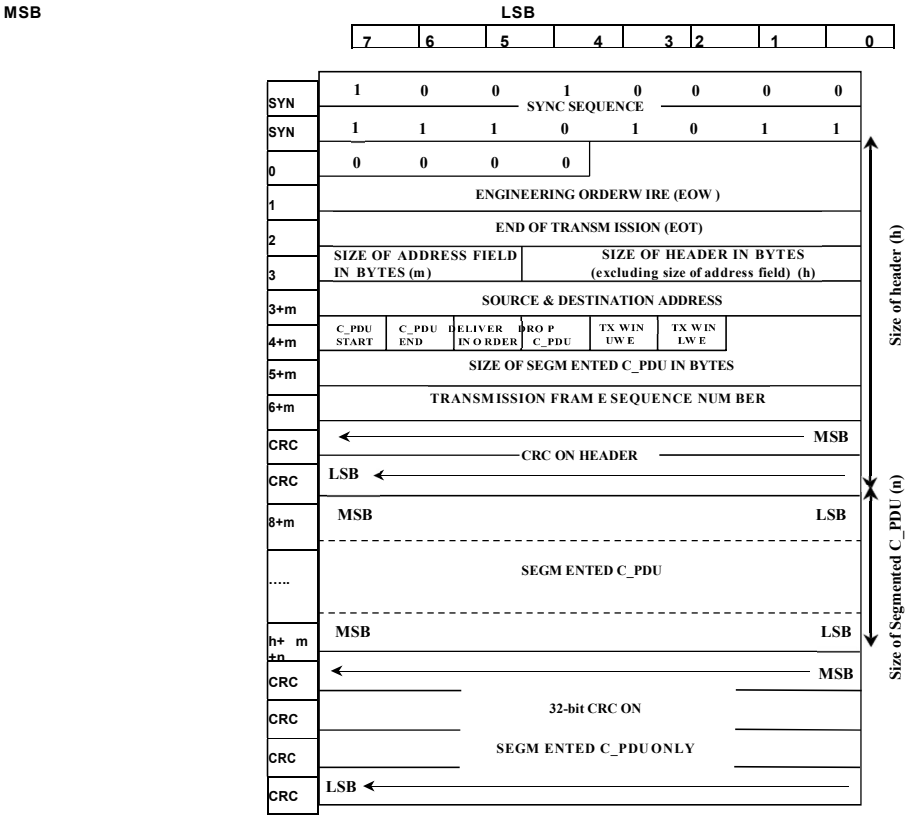


Figure C-9. Frame format for ED3-DATA-ONLY D_PDU Type 0

The encoding of the ED3-DATA-ONLY D_PDU is shown in Figure C-9.

C.4.4.2. DATA-ONLY (Type 9) D_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	DPDU Type = 9				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	C_PDU START	C_PDU END	DELIVER IN ORDER	DROP C_PDU	TX WIN UWE	TX WIN LWE		
5+m	Segment Size (n)							
6+m*	Transmit Frame Sequence Number (increased 8 to 16 bits)							MSB
7+m*	LSB							
CRC	CRC on Header							
CRC								
h+m	Segmented C_PDU							
h+m+n-1								
CRC	CRC on Data							
CRC								
CRC								
CRC								

Figure C-10. Frame format for DATA-ONLY D_PDU Type 9

The encoding of the DATA-ONLY D_PDU is shown in Figure C-10. The differences relative to ED3-DATA-ONLY D_PDU are highlighted and numbering changes marked with "*".

C.4.4.3. DATA-ONLY Fields

The term "DATA-ONLY D_PDU" in this section is used to refer to both Type 0 and Type 9 D_PDUs.

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, Figure C-10, 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.5

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.5

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.

The DROP C_PDU flag is used when the TTL of a C_PDU expires before transmission of the C_PDU is complete. It is necessary to send and acknowledge this D_PDU in order to maintain window synchronization. A D_PDU with the DROP C_PDU flag set **shall** be acknowledged.

The Segmented C_PDU field **shall** be empty if the DROP C_PDU flag is set as the data is being discarded and the SIZE OF SEGMENTED C_PDU field **shall** (10) be zero in this case.

When the DROP C_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.

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.4.2.10.

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
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. This integer **shall** be modulo 256 for Type 0 D_PDU and modulo 65,536 for type 9.

Values for the TRANSMIT FRAME SEQUENCE NUMBER field **shall** (16) be assigned in an ascending order of appropriate modulo 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.4.2.9 .

C.4.5. ACK-ONLY (TYPE 1 or TYPE 10) D_PDU (Acknowledgement of type 0, 2 data transfer)

C.4.5.1. ED3-ACK-ONLY (Type 1) D_PDU

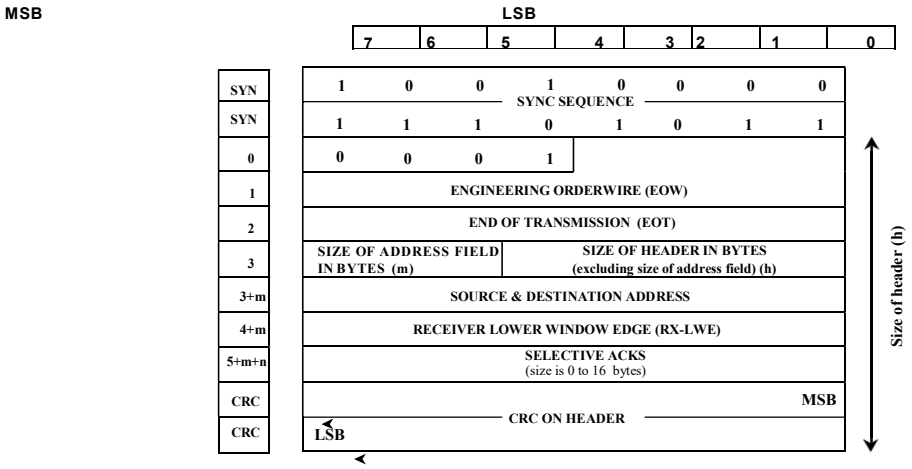


Figure C-11. Frame format for ED3-ACK-ONLY D_PDU Type 1

The encoding of the ED3-DATA-ONLY D_PDU is shown in Figure C-11.

C.4.5.2. ACK-ONLY (Type 10) D_PDU

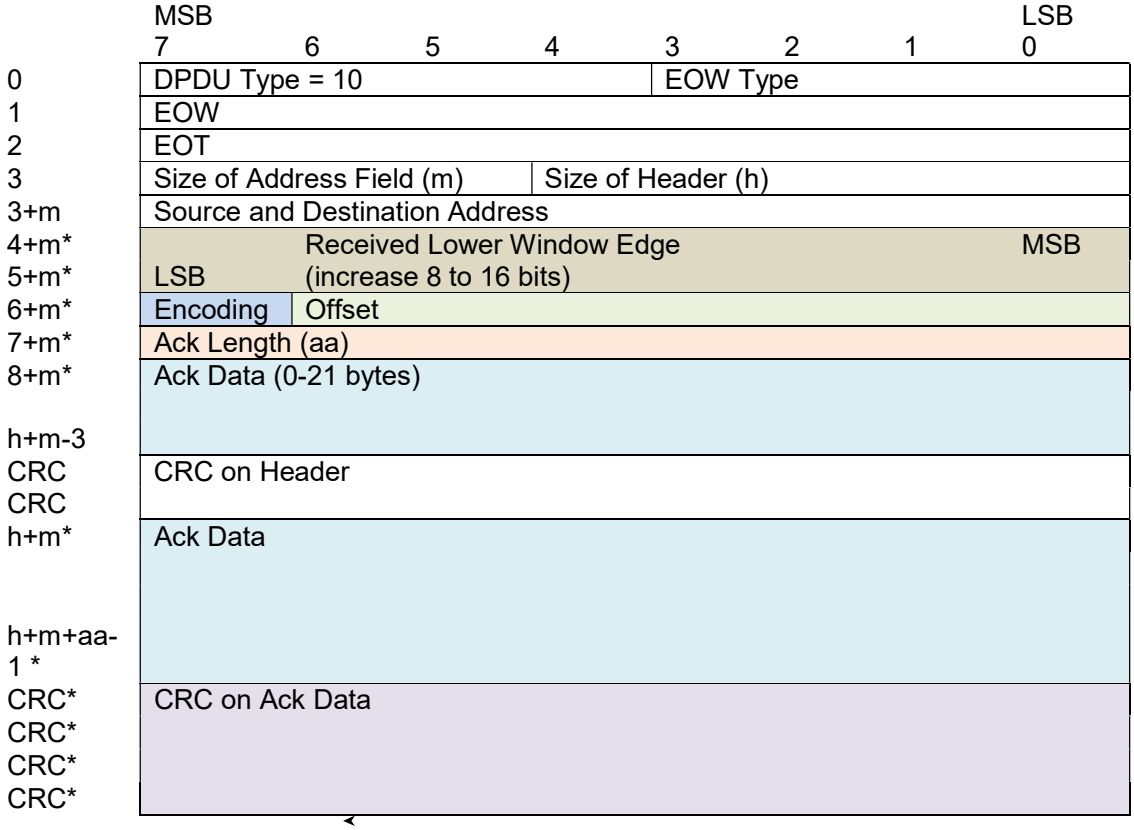


Figure C-12. Frame format for ACK-ONLY D_PDU Type 10

The encoding of the ACK-ONLY D_PDU is shown in Figure C-12. The differences relative to ED3-ACK-ONLY D_PDU are highlighted and numbering changes marked with "*".

C.4.5.3. ACK-ONLY D_PDU Common Encoding

The term "ACK-ONLY D_PDU" in this section is used to refer to both Type 1 and Type 10 D_PDUs.

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 or (Edition 4 only) when necessary acknowledgement information cannot be encoded in DATA-ACK D_PDU.

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-11 or Figure C-12

and the paragraphs below:

- RECEIVE LOWER WINDOW EDGE (LWE)

The value of the RECEIVE LOWER WINDOW EDGE (RX LWE) field **shall** (3) equal the D_PDU sequence number of the RX LWE pointer associated with the node's receive ARQ flow-control window**.** This integer **shall** be modulo 256 for Type 1 D_PDU and modulo 65,536 for type 10.

C.4.5.4. ED3-ACK-ONLY D_PDU Encoding for Type 1 only

The ED3-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-11 and the paragraphs below:

- SELECTIVE ACKS

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 ED3-ACK-ONLY D_PDU and the value of the Size of Header field, and is not provided explicitly in the ED3-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, except that 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-13, Figure C-14 and the paragraphs below.

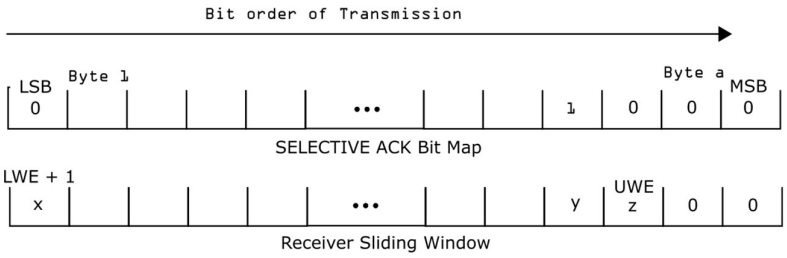


Figure C-13. Constructing the SELECTIVE ACK Field
(Note: the bits that are set (1) and cleared (0) are representing example bits)

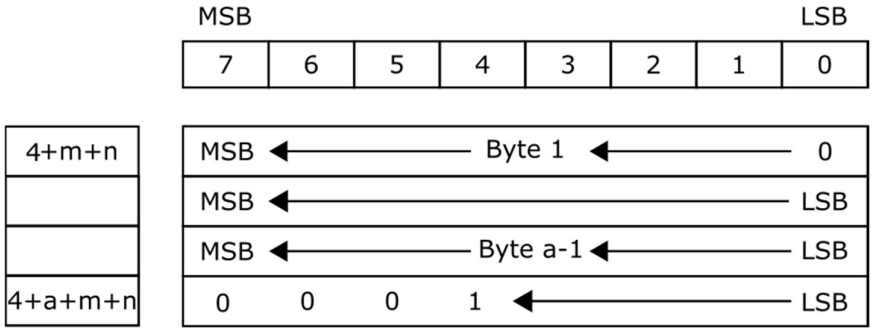


Figure C-14. SELECTIVE ACK mapping convention
(Note: the example bits that were shown in Figure C-13 can been seen in this mapping example)

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 ED3-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.4.5.5. ACK-ONLY D_PDU Encoding for Type 10 only

The Ack Data in the ACK-ONLY D_PDU encodes acknowledgements of transmitted D_PDU frames. In order to efficiently encode acknowledgements when the window size is large, two different encodings are defined. Acknowledgements **may** be encoded in multiple ACK-ONLY D_PDUs in order to keep the size of each D_PDU smaller. It is recommended to keep ACK_ONLY D_PDU size small in order to reduce risk of D_PDU loss during transmission. It is also recommended to transmit ACK_ONLY D_PDUs multiple times, because loss of acknowledgement information can lead to delays and unnecessary retransmission.

When a set (1 or more) of ACK-ONLY D_PDUs is sent all of the received DATA PDUs **shall** be acknowledged. The number and encoding of these D_PDUs is an implementation choice.

Only D_PDU frames with a correct segmented C_PDU CRC **shall** (6) be acknowledged positively even if CRC is correct, except that frames with the DROP C_PDU flag set **shall** (7) be acknowledged positively the header regardless of the results of the CRC check on the segmented C_PDU.

If Ack Data can be encoded in 21 bytes or less the D_PDU encoding of Figure C-1(a) is used. All of the Ack Data is encoded in the D_PDU header.

If Ack Data is larger, the encoding with two CRCs of Figure C-1(b) is used. In this case no Ack data **shall** be encoded in the header. All of the Ack Data **shall** be encoded in the main D_PDU after the header.

The maximum window size can lead to up to 32,768 packets to be acknowledged, which would need up to 4096 bytes with the standard encoding mechanism. It is generally be desirable to keep ACK-ONLY D_PDUs smaller than this, in order to reduce risk of D_PDU loss. The offset mechanism provides a means to do this, by enabling reporting acknowledgements by referencing D_PDUs relative to an offset of the Lower Window Edge. Multiple ACK-ONLY D_PDUs **may** be sent with different offsets, in order to acknowledge all D_PDUs received. The Offset is encoded as 7 bits in the ACK-ONLY D_PDU header. The Offset value is multiplied by 256 to determine the size of the offset in bits (where each bit represents a PDU that is being acknowledged). This enables the acknowledgement data to be reduced, so that the Ack Data **may** be reduced so that it will never exceed 32 bytes.

The size in bytes of Ack Data used is specified in Ack Length encoded in the ACK-ONLY D_PDU header. It **may** be up to 255 bytes.

Two options are provided to encode Ack Data, with the choice controlled by the Encoding Bit in the ACK-ONLY D_PDU header. A sender **may** choose to use either or both encodings. A recommended approach is for the sender to evaluate both encodings and then to use the more compact one. A receiver **shall** support both

encodings.

The first mechanism is the one defined in Section C.4.5.4. The Encoding bit is set to 0 for use of this encoding. This encoding sets a bit to 1 to indicate that a Data PDU has been accepted. The data starts with reference to the first PDU after the Offset. Bit encoding follows the rules of Section C.4.5.4. It is safe to add trailing zeros to pad to an exact number of bytes.

The second is a simple Run Length encoding mechanism described in this section. It is designed as a byte-aligned format that can efficiently encode long runs of 1s or 0s, which are anticipated to be likely when the window size is large.

Run Length Encoding starts with the standard Ack encoding defined in Section C.4.5.4 and generates a different encoding. In some common situations, this will be more compact. It is recommended to use the more compact encoding. The encoding is illustrated in Figure C-15 and Figure C-16.

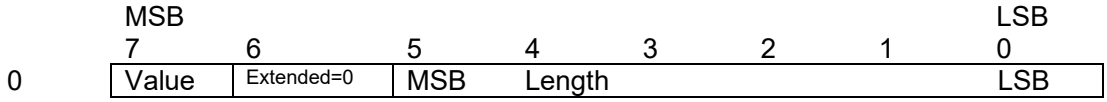


Figure C-15. Run Length Encoding: Single Byte Encoding

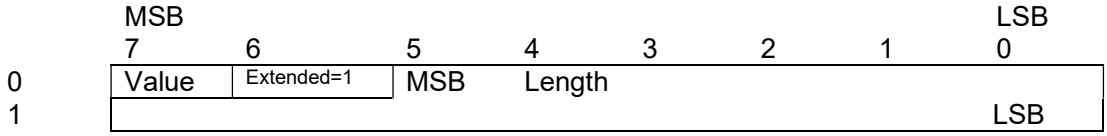


Figure C-16. Run Length Encoding: Two Byte Encoding

The first bit (Value) is set to 0 or 1 and shows the value that is being repeated. If the Extended bit is set to 0, the encoding uses one byte and Length is specified in 6 bits. If the Extended bit is set to 1, the encoding uses two bytes and Length is specified in 14 bits. The Length indicates the number of repeats of the value, so if length is zero the value occurs once and is not repeated.

These encodings are repeated through Ack Data.

C.4.6. DATA-ACK D_PDU

C.4.6.1. ED3-DATA-ACK (Type 2) D_PDU

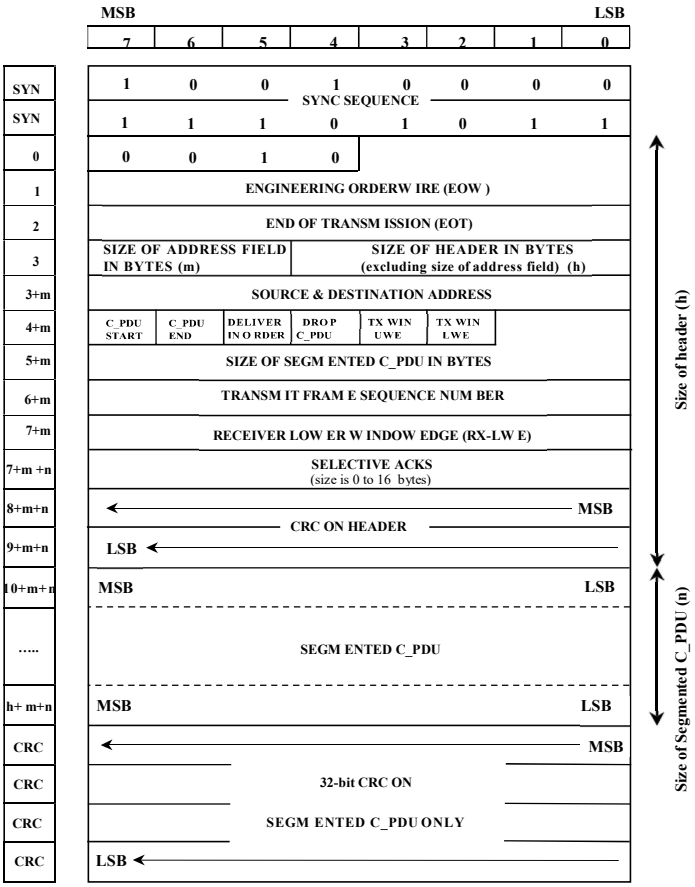


Figure C-17. Frame format for ED3-DATA-ACK D_PDU Type 2

The encoding of the ED3-DATA-ACK D_PDU is shown in Figure C-17.

C.4.6.2. DATA-ACK (Type 11) D_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	DPDU Type = 11				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	C_PDU START	C_PDU END	DELIVER IN ORDER	DROP C_PDU	TX WIN UWE	TX WIN LWE		
5+m	Segment Size (n)							
6+m*	Transmit Frame Sequence Number (increase 8 to 16 bits)							MSB
7+m*	LSB							
8+m*	Received Lower Window Edge (increase 8 to 16 bits)							MSB
9+m*	LSB							
10+m*	Encoding	Not Used						
11+m*	Selective Acks (0-18) bytes (increase from 16 to 18 bytes)							
…. h+m-3								
CRC CRC	CRC on Header							
h+m	Segmented C_PDU							
h+m+n-1								
CRC CRC CRC CRC	CRC on Data							

Figure C-18. Frame format for DATA-ACK D_PDU Type 11

The encoding of the DATA-ACK D_PDU is shown in Figure C-18. The differences relative to ED3-DATA-ACK D_PDU are highlighted and numbering changes marked with "*".

C.4.6.3. DATA-ACK D_PDU Common Encoding

The term "DATA-ACK D_PDU" in this section is used to refer to both Type 2 and Type 11 D_PDUs.

DATA-ACK D_PDU **shall** be used when data and acknowledgements are being transmitted, rather than DATA-ONLY plus ACK-ONLY D_PDUs. If only one acknowledgement is being sent, use of DATA_ACK D_PDU is preferred as overhead is reduced. All of the acknowledgement information is in the header, so the

acknowledgement information can be valid when data is corrupted in transfer. If acknowledgements are repeated, it is recommended to use ACK-ONLY D_PDUs to do this.

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 D_PDU or a DATA-ACK 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-17 and the referenced paragraphs:

- C_PDU START – **shall** be as specified in Section C.4.4.3 for the DATA-ONLY D_DPU;
- C_PDU END– **shall** be as specified in Section C.4.4.3.C.4.4 for the DATA-ONLY D_DPU;
- DELIVER IN ORDER– **shall** be as specified in Section C.4.4.3 for the DATA-ONLY D_DPU;
- DROP C_PDU– **shall** be as specified in Section C.4.4.3 for the DATA-ONLY D_DPU;
- TX WIN UWE– **shall** be as specified in Section C.4.4.3C.4.4 for the DATA-ONLY D_DPU;
- TX WIN LWE– **shall** be as specified in Section C.4.4.3 for the DATA-ONLY D_DPU;
- SIZE OF SEGMENTED C_PDU– **shall** be as specified in Section C.4.4.3C.4.4 for the DATA-ONLY D_DPU;
- TRANSMIT SEQUENCE NUMBER– **shall** be as specified in Section C.4.4.3 for the DATA-ONLY D_DPU;
- RECEIVE LOWER WINDOW EDGE (RX LWE) – **shall** be as specified in Section C.4.5.3 for the ACK- ONLY D_DPU;

C.4.6.4. Encoding Selective Acks

For ED3-DATA-ACK (Type 2), SELECTIVE ACKS **shall** be as specified in Section C.4.5.4 for the ED3-ACK-ONLY D_DPU.

For DATA-ACK(Type 2), a subset of the rules of Section C.4.5.5 shall be followed as described here. The Encoding bit controls the choice of encoding in the same

manner as in Section C.4.5.5.

Selective Acks **shall** be encoded in the manner of the Ack Data encoding set out in Section C.4.5.5, controlled by the encoding choice. Selective Acks is limited to 18 bytes, which may prevent acknowledgement of D_PDUs a long way from the receiving window edge. In this situation, ACK-ONLY D_PDU **shall** be used to acknowledge these D_PDUs.

C.4.7. RESET/WIN-RESYNC D_PDU (Reset/Re-synchronise peer protocol entities)

C.4.7.1. ED3-RESET/WIN-RESYNC (TYPE 3) D_PDU

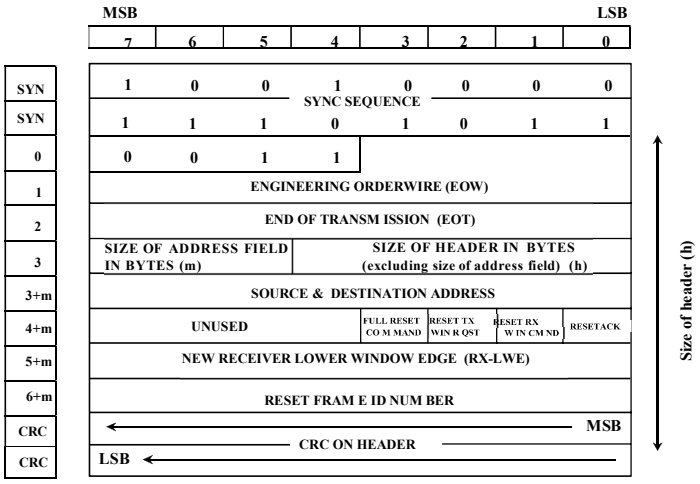


Figure C-19. Frame format for ED3-RESET/WIN-RESYNC D_PDU Type 3

The encoding of the ED3-RESET/WIN_RESYNC D_PDU is show in Figure C-19.

C.4.7.2. RESET/WIN-RESYNC (TYPE 12) D_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	DPDU Type = 12				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	Unused				FRC	RTWQ	RRWC	RSA
5+m*	New Receiver Lower Window Edge							MSB
6+m*	LSB (increase 8 to 16 bits)							
7+m	Reset Frame ID Number							
CRC	CRC on Header							
CRC								

Figure C-20. Frame format for RESET/WIN-RESYNC D_PDU Type 12

The encoding of the RESET/WIN-RESYNC D_PDU is shown in Figure C-10Figure C-20. The differences relative to ED3- RESET/WIN-RESYNC D_PDU are highlighted and numbering changes marked with "*".

C.4.7.3. RESET/WIN-RESYNC Fields

The term "RESET/WIN-RESYNC D_PDU" in this section is used to refer to both Type 3 and Type 12 D_PDUs.

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.7.2.

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, ACK, and DATA-ACK 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-19 and the paragraphs below:

- FULL RESET COMMAND
- RESET TX WIN RQST
- RESET RX 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.7.2 and C.7.3.

A RESET/WIN-RESYNC 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 RESET/WIN-RESYNC 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 RX WIN CMND flag, as specified below.

A RESET/WIN-RESYNC D_PDU with the RESET RX WIN CMND flag set equal to one (1) **shall**
be used to force a resynchronisation of the TX-LWE pointer at the sending node and the RX-LWE pointer at the receiving node.

A node that sends a RESET/WIN-RESYNC D_PDU with the RESET RX 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 TX-LWE.
- The sending node **shall** (13) wait for a RESET/WIN-RESYNC 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 RESET/WIN RESYNC D_PDU with the RESET RX 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 D_PDU with the RESET ACK flag set equal to one as an acknowledgement that the resynchronization has been performed.

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

A node may use a RESET/WIN-RESYNC acknowledgement transmission, i.e., a RESET/WIN-RESYNC D_PDU with the RESET ACK 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 two way data 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 RX 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 of appropriate modulo 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.

C.4.8. **EXPEDITED-DATA-ONLY (TYPE 4) D_PDU**

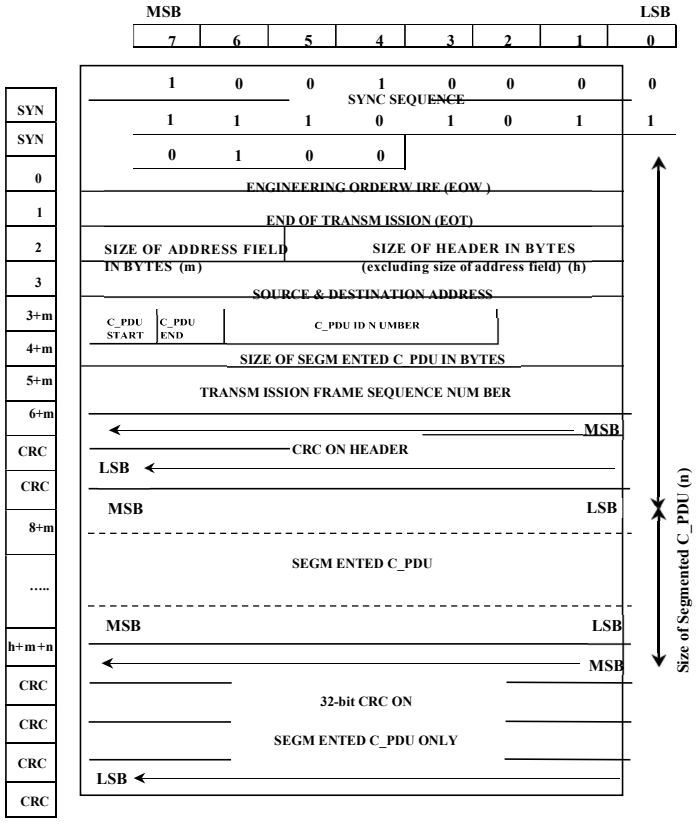


Figure C-21. Frame format for EXPEDITED DATA ONLY D_PDU Type 4

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** send an EXPEDITED-ACK-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 ED3-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-21 and the paragraphs noted:

C_PDU START – **shall** (4) be as specified for the DATA-ONLY D_PDU in Section C.4.4

C_PDU END – **shall** (5) be as specified for the DATA-ONLY D_PDU in Section C.4.4
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.4.2.10 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.4.4, 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 ED3-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-21. 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 two way data 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.4.9. **EXPEDITED-ACK-ONLY (TYPE 5) D_PDU (Acknowledgement of expedited data transfer)**

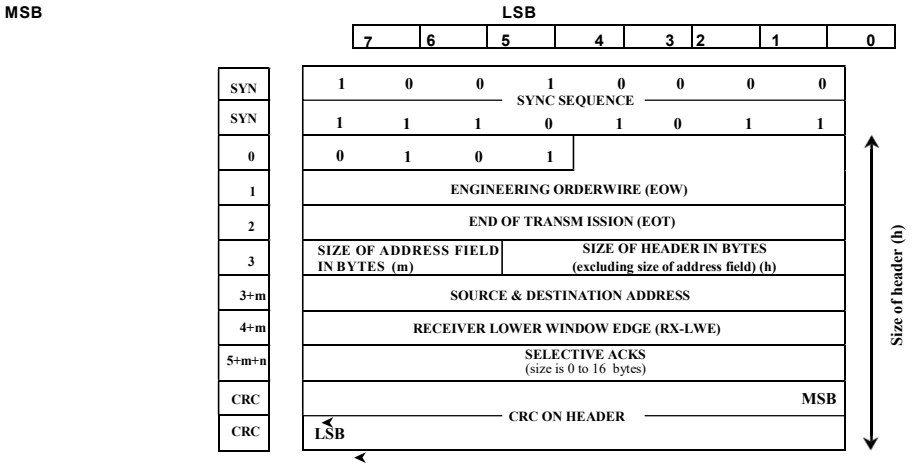


Figure C-22. Frame format for EXPEDITEDACK-ONLY D_PDU Type 5

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 ED3-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-22.

C.4.10. **MANAGEMENT (TYPE 6) D_PDU (Management message transfer)**

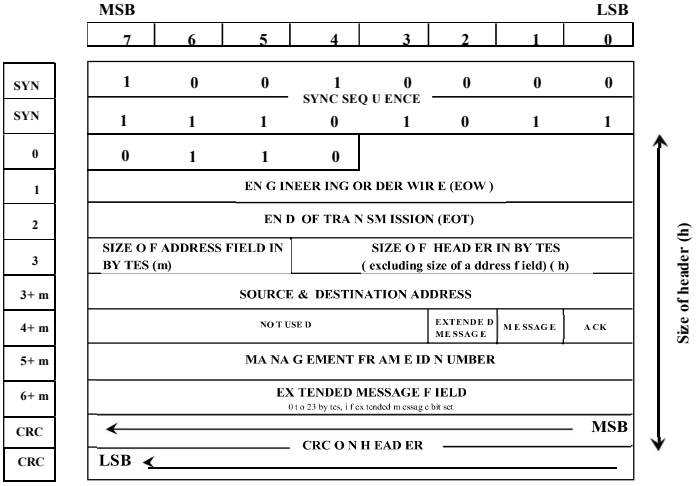


Figure C-23. Header format for MANAGEMENT D_PDU TYPE 6

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.
MANAGEMENT D_PDU transmission bypasses other D_PDUs, so that they can be delivered as quickly as possible. Repeating transmission of MANAGEMENT D_PDUs is recommended.

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-23 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.

Management D_PDUs that have been repeated **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** 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).

The 12-bit EOW section of the D_PDU **shall** (27) carry the EOW (non-extended) MANAGEMENT message, as specified in Section C.6.

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.4.11. **NON-ARQ-DATA (TYPE 7) D_PDU (Non-ARQ data transfer)**

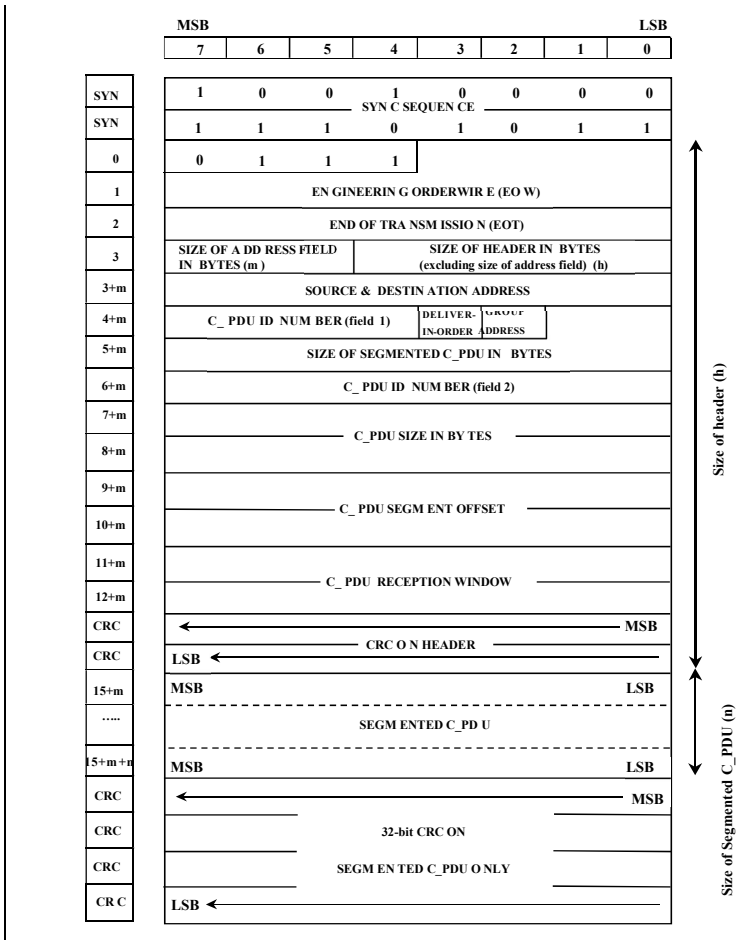


Figure C-24. Frame format for NON-ARQ-DATA D_PDU TYPE 7

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-24 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-25.

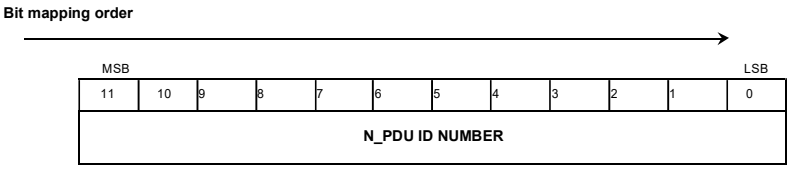


Figure C-25. C_PDU ID NUMBER Field

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-26:

- 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).

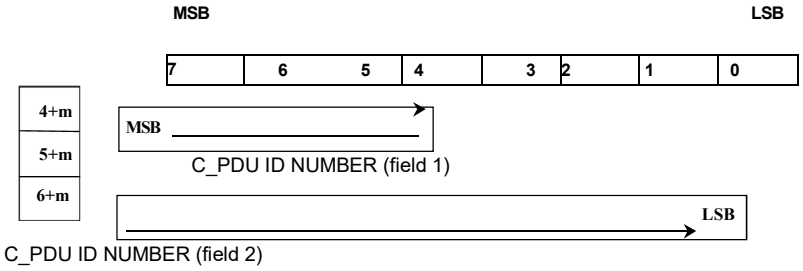


Figure C-26. C_PDU ID NUMBER mapping convention in D_PDU header

The DELIVER IN ORDER flag **shall not** be set. The bit field is retained for compatibility with Edition 3. Operational experience has shown that in order delivery with the NON-ARQ service is problematic. With Non-ARQ in order delivery is expected as transfer is in order.

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

1) C_PDU is complete and error free; or
2) The C_PDU RECEPTION WINDOW expires. This can be helpful to partial C_PDU delivery.

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.4.2.10.

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-27. 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-28.

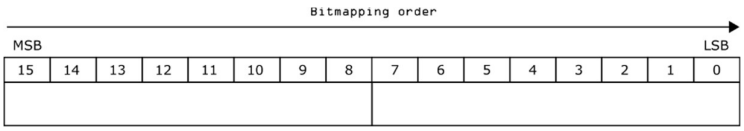


Figure C-27. C_PDU SIZE Field

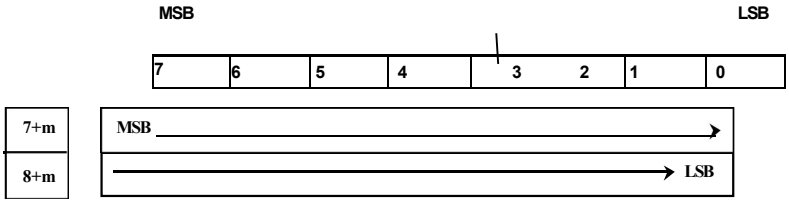


Figure C-28. C_PDU SIZE mapping convention in D_PDU header

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-29. 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-30.

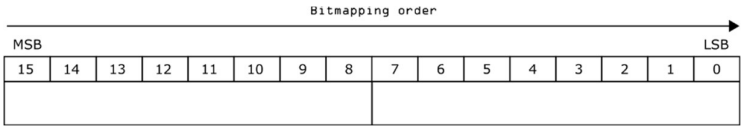


Figure C-29. C_PDU SEGMENT OFFSET Field

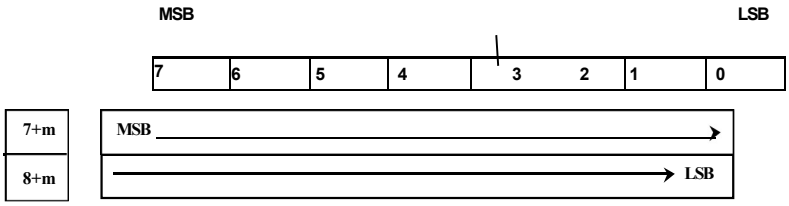


Figure C-30. C_PDU SEGMENT OFFSET mapping convention in D_PDU header

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.

Setting this value needs some care, as if set too large it can delay delivery of partial C_PDUs. If set to low, it will miss valid C_PDU segments.

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-31. 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-32.

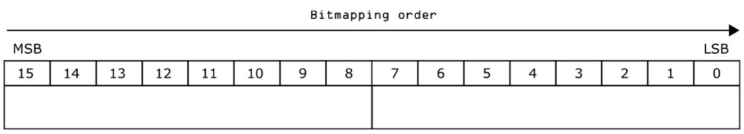


Figure C-31. C_PDU RECEPTION WINDOW Field

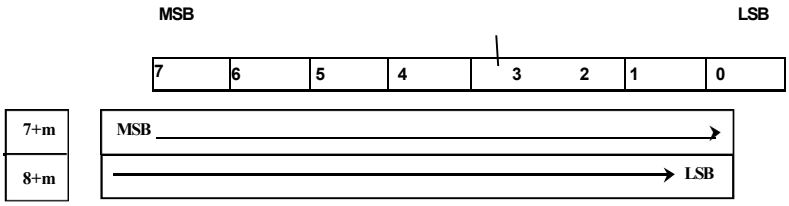


Figure C-32. C_PDU RECEPTION WINDOW mapping convention in D_PDU header

C.4.12. EXPEDITED-NON-ARQ-DATA (TYPE 8) D_PDU (Expedited non-ARQ data transfer)

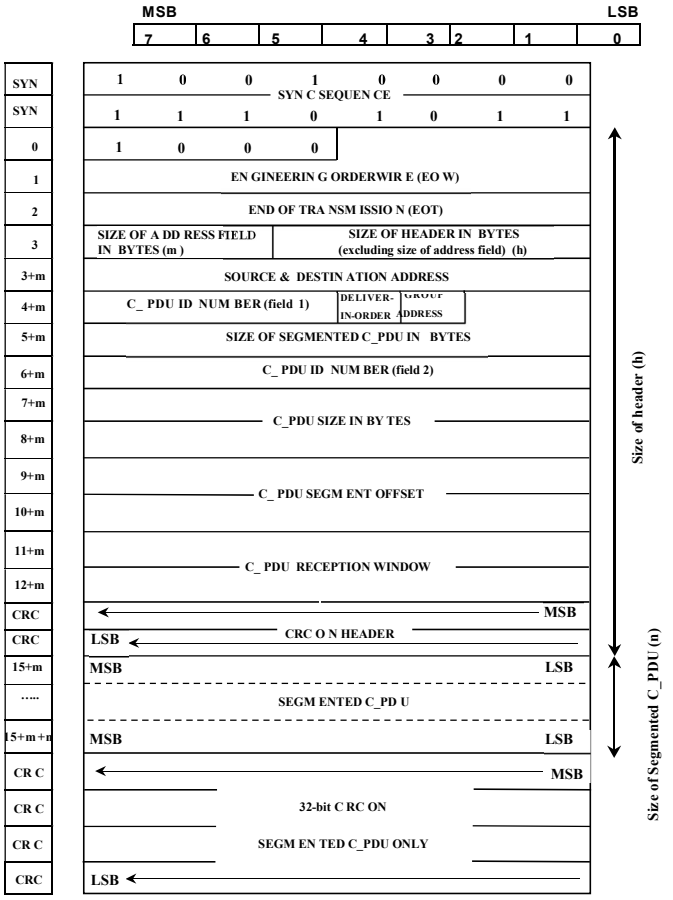


Figure C-33. Frame format for EXPEDITED-NON-ARQ DATA D_PDU Type 8

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-33.

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.4.13. **EXTENSION (TYPE 13) D_PDU**

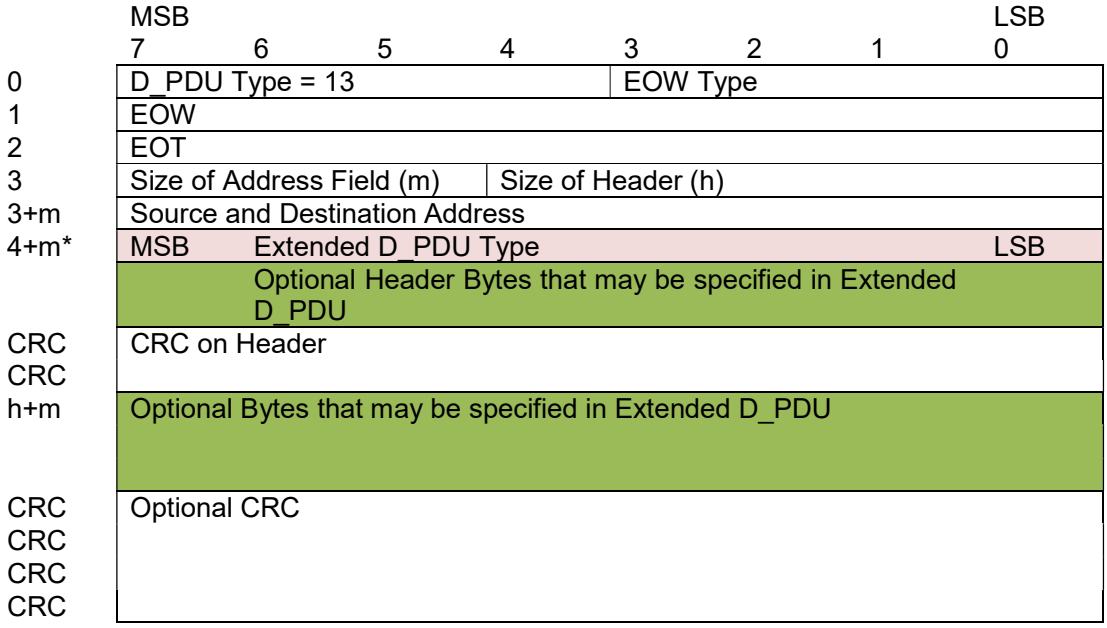


Figure C-34. Frame format for EXTENSION D_PDU Type 13

All of the D_PDU Type values are assigned in this edition of Annex C. The EXTENSION D_PDU (TYPE 13) D_PDU enables future versions of this Annex and other Annexes of STANAG 5066 to define new D_PDU types. A list of assigned Extended D_PDU types is set out in Section C.8.

The EXTENSION D_PDU **shall** only be used with Edition 4 (or subsequent) peers.

The EXTENSION D_PDU format is shown in Figure C-34. The Extended D_PDU Type field Extended D_PDU Type allows for up to 256 new D_DPDUs to be defined. Extended D_PDU can have two formats:

1. Header only format, as specified in Figure C-2(a), which ends with two byte CRC on header.
2. The format specified in Figure C-2(b), which had data beyond the header, ending with a four byte checksum on the data beyond the header.

The choice of format **shall** be specified for each Extended D_PDU Type. The format choice **may** vary based on information in the D_PDU header.

C.4.14. PADDING (TYPE 14) D_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 14				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
CRC	CRC on Header							
CRC								

Figure C-35. Frame format for PADDING D_PDU Type 14

PADDING D_PDU **shall** only be used with Edition 4 (or subsequent) peers.

The format of the PADDING D_PDU (Type 14) is shown in Figure C-35. This is a minimal D_PDU with no fields specific to the D_PDU type.

The DTS sends out data as stream of D_PDUs. Modern HF Waveforms send data in blocks, which means that the volume of data sent needs to be an exact number of block sizes. There is generally some extra space for one of two reasons:

a. There is insufficient space to fit in the next D_PDU (often a DATA-ONLY D_PDU which might typically by 2-300 bytes); or
b. Only a small amount of data is being sent, and there is space.

This annex allows D_PDUs to be repeated, which is preferable to padding with non-DPDU data. Benefits of repeating D_PDUs:

1. If you repeat a D_PDU, risk of data loss is reduced as the original may be corrupted.
2. An EOW (Engineering Order Wire) single byte message can be sent with each D_PDU to exchange control information. Extra D_PDUs allow for more EOWs to be sent which allows for more information to be sent and reduces the risk of (important) EOW information from being lost.
3. The EOT (End of Transmission) is a single byte sent with each DPDU to indicate time of transmission remaining. This allows the receiver to determine the length of a transmission, which will enable it to know when the transmission is complete and when it can start transmission. This is very important for resilient operation. Transmitting more EOTs reduces the risk of all EOT information getting lost.

EOT and EOW repeats are particularly important at slower speeds when only a small number of D_PDUs will be sent in each transmission. It will generally make sense to repeat critical data. ACK-ONLY D_PDUs are particularly useful to repeat and are small. It can also make sense to repeat high priority D_PDUs, giving priority to SAPs

with low latency QoS requirements. Note that when a physical link has been established (CAS-1 for ARQ traffic) that ACK-ONLY D_PDSs can be used in both directions which is recommended.

The PADDING D_PDU is provided as a minimal D_PDU to gain the second and third advantages of repeating D_PDUs, when it is not possible to send another D_PDU because all the valid options are too large. This is particularly likely to happen for non-ARQ traffic, where it is not possible to send an ACK-ONLY D_PDU. It can also happen where the ACK-ONLY D_PDUs available to transmit are too large.

The PADDING D_PDU is encoded following the generic D_PDU structure as shown in Figure C-2(a) with the D_PDU Type set to 14. There are no D_PDU specific fields for the PADDING D_PDU.

Note that EOWs in a padding D_PDU are directed to the node or nodes indicated by the destination address, which may be a broadcast address.

C.4.15. **WARNING (TYPE 15) D_PDU**

The WARNING D_PDU is sent in response to unexpected or unrecognised D_PDU type.

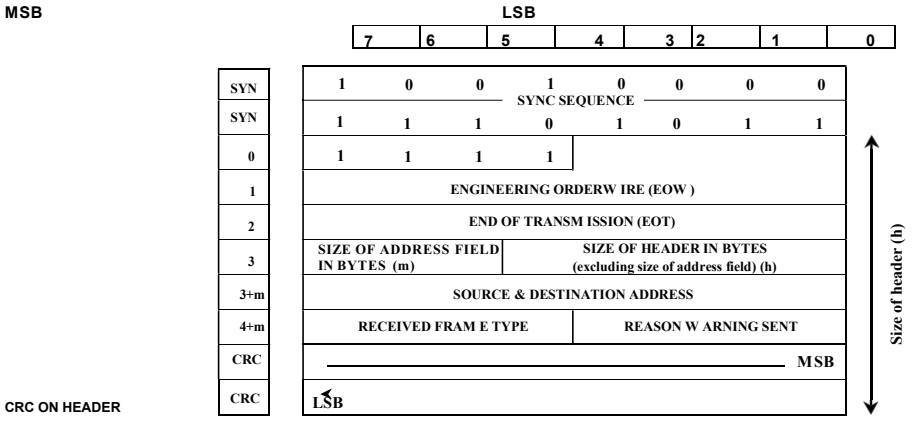


Figure C-36. Frame format for WARNING D_PDU Type 15

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-36 and the paragraphs below:

- RECEIVED FRAME TYPE
- REASON WARNING 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-37 and Figure C-38.

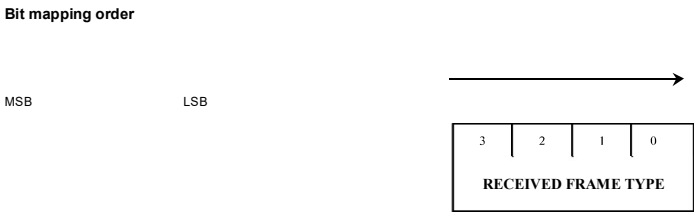


Figure C-37. RECEIVED FRAME TYPE Field

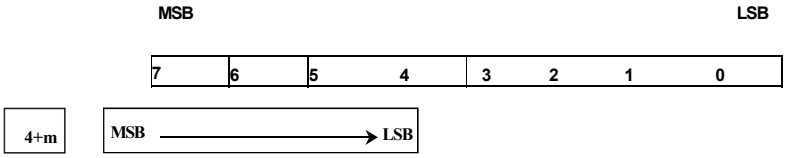


Figure C-38. RECEIVED FRAME TYPE mapping convention in D_PDU header

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

Reasonn	Field
Unrecognised D_PDU type	0
Connection-related D_PDU Received While Not Currently	1
Invalid D_PDU Received	2
Invalid D_PDU Received for Current State	3
Unrecognized D_PDU Extension Type	4
Unspecified/reserved	5-15

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-39 and Figure C-40. Note that as all D_PDU values are used in Edition 4, reason value 0 is only expected from Edition 3 peers.

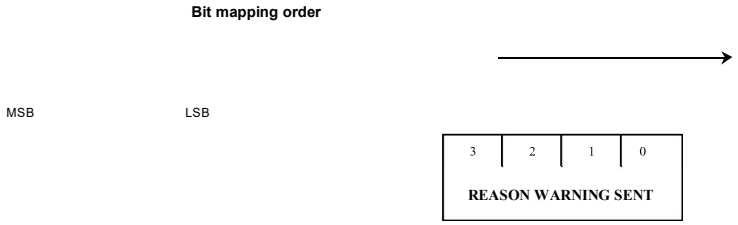


Figure C-39. REASON WARNING SENT Field

Figure C-40. 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)
5. A node receives a D_PDU header addressed to itself with a valid CRC and an unrecognized Extended D_PDU Type (value 0100)

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

C.5. 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.5.1. ARQ Mode Segmentation and Re-assembly

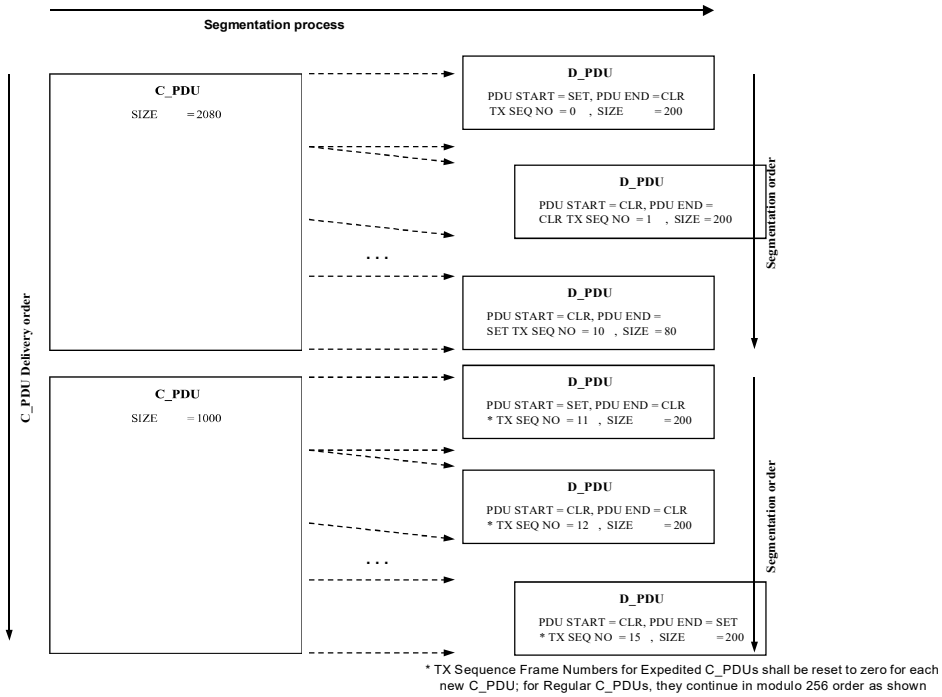


Figure C-41. C_PDU Segmentation for ARQ Delivery Services (Regular and Expedited Service)

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-41 and as follows:

a. The Maximum C_PDU-Segment Size within a D_PDU for ARQ-Delivery services **shall** 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.]

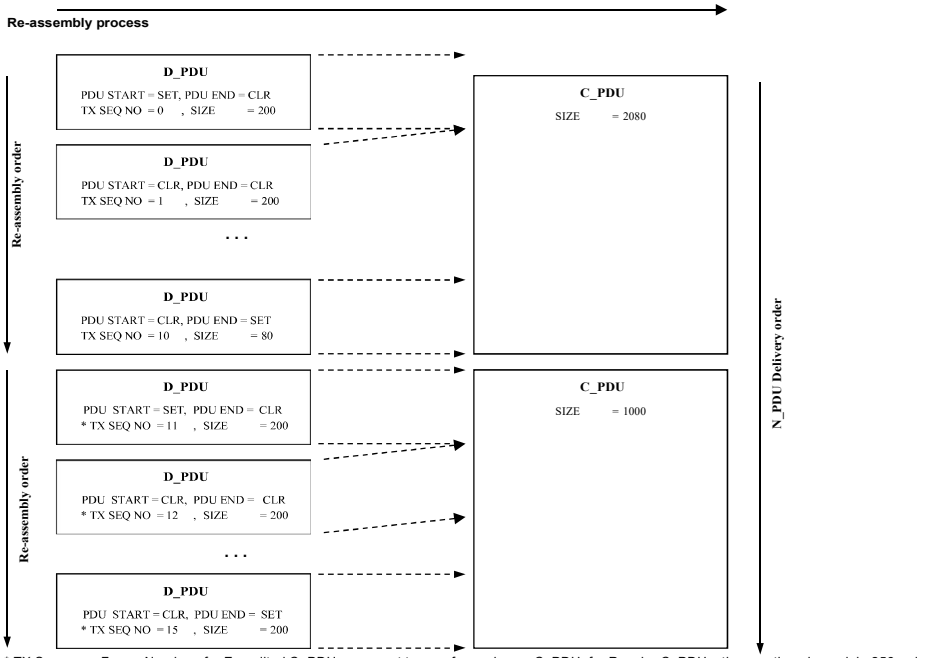


Figure C-42. C_PDU Re-assembly for ARQ Delivery Services (Regular and Expedited Service)

Re-assembly of a C_PDU from its segments **shall** (9) be performed in accordance with the example shown in Figure C-42 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.5.1.1. Notes on ARQ Selection of Maximum C_PDU Segment Size

The best choice of Maximum C_PDU Segment Size is related to the choice of transmission speed and interleaver. This is an implementation choice. Often, particularly at lower HF speeds, it will be desirable to optimize throughput. When optimizing throughput, high Frame Error Rates (FER) which may be around 50% FER will lead to best throughput. At higher HF and WBHF speeds, it will sometimes be preferable to optimize for low latency, leading to selection of a more conservative transmission speed.

When optimizing for throughput, the choice of Maximum C_PDU Segment Size is critical. If too small a size is chosen, the overhead of D_PDU headers is too large. If too large a size is chosen, the overhead of retransmission due to frame errors is too large.

STANAG 5066 Edition 3 Annex H (Implementation Notes and Guidance) notes various research on measurements at 600 – 1200 bps, looking to optimize throughput. A key observation is that a Maximum C_PDU Segment Size of 200 bytes is a “good compromise”.

It is also noted “If there is a desire to vary frame size in some way, STANAG 5066 supports the use of variable frame sizes. While the data available to date suggest that the benefit may be marginal, it would be possible, for example, to associate a certain frame size with each data rate.”

More recent observations on this choice have been made, which suggest:

- At mid-range narrowband HF speeds, a Maximum C_PDU Segment Size of 200-300 bytes is a good choice. This is in line with the earlier observations.
- When optimizing for throughput, long transmissions give the best performance.
- Intermediate Term Variation (10-120 secs) gives key impact on choice of Maximum C_PDU Segment Size.
- At 75 or 150 bps, a Maximum C_PDU Segment Size of 100 bytes is a good

choice.

- At speeds increase into WBHF range increasing Maximum C_PDU Segment Size to larger values up to the maximum of 1023 bytes is optimal.
- For optimal 1023-byte segmentation, an optimized MTU value is recommended: With an MTU of 2048 bytes, 5 bytes are added by SIS and 1 byte is added by CAS, which results in 2054 byte Data C_PDU. This will result in 2x1023 byte C_PDU segments, and a third 2-byte C_PDU segment. Therefore an MTU of 2042 bytes is recommended to optimally fill 2x1023 byte segments

Collectively, these results suggest that selection of Maximum C_PDU Segment Size based on anticipated data rate is desirable.

C.5.2. **NON-ARQ Mode Segmentation and Re-assembly**

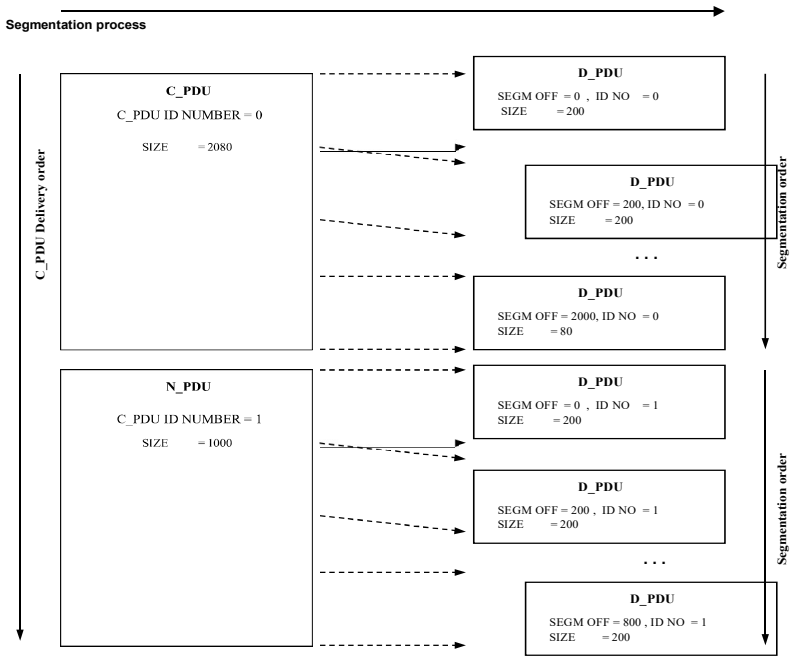


Figure C-43. C_PDU Segmentation for Non-ARQ Delivery Services (Regular and Expedited Service)

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-43, 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.4.11;

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.

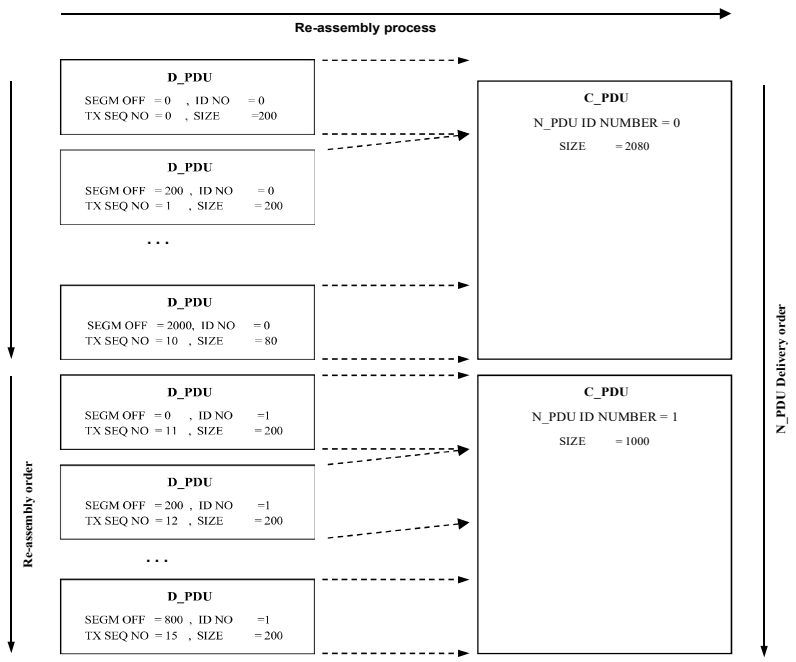


Figure C-44. C_PDU Re-assembly for Non-ARQ Delivery Services (Regular and Expedited Service)

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-44 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) C_PDUs shall be delivered in the order they arrive. Deliver-In-Order Mode shall not be supported for this edition of STANAG 5066.

e) 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.5.2.1. Non-ARQ D_PDU Ordering and Priority

For ARQ service, C_PDU fragments **shall** be transmitted in order with sequential frame sequence numbers. There is no equivalent requirement for non-ARQ, a

C_PDU fragments use an independent identifier for correlation.

When transmitting a sequence of non-ARQ C_PDU fragments, it **may** be necessary or desirable to insert other D_PDUs in between them, in particular higher priority D_PDUs arriving after transmission began. Note that this will affect the reception window that was calculated when the first part of a Non-ARQ D_PDU was sent and **may** result in timeout, resulting in data loss

C.5.2.2. Notes on NON-ARQ Selection of Maximum C_PDU Segment Size

When using NON-ARQ it will generally be desirable to use more conservative transmission rates than for ARQ. In some cases (e.g., broadcast), errors will not be corrected and in other use cases correction will be controlled by application timers which will take longer. Because of this, it is important that data loss is low, and this means low transmission rate.

A related consideration is that Non-ARQ allows the sender to request multiple transmissions. If data is sent only once, there no benefit arising from reducing C_PDU Segment Size and a large value (1023 bytes) can be used to reduce overhead. When there are repeats, a smaller Maximum C_PDU Segment Size may be beneficial to minimizing data loss.

For applications using Non-ARQ with errors, smaller Maximum C_PDU Segment Size is potentially beneficial to give the applications error-free fragments. There is little operational experience with such applications.

Note the considerations for Non-ARQ applications are significantly different and can vary between the different classes of Non-ARQ applications noted below. A clear Quality of Service model is desirable to support this.

C.5.2.2.1. Considerations for Broadcast Applications

Broadcast applications will either have very slow error handling or none at all for EMCON reception. It is therefore important to choose transmission speed, interleaver, Non-ARQ repeat count and Maximum C_PDU Segment Size with care.

For EMCON applications, it may be appropriate to consider use of non-ARQ with errors.

C.5.2.2.2. Considerations for Multicast Applications

For multicast applications, such as ACP 142, there will be application level retransmission of errors and/or some tolerance of loss due to application-level Forward Error Correction.

It is anticipated that for such applications that transmissions rate will be chosen so that base frame error rate is low and that repeating the Non-ARQ transmissions will

be sub-optimal. Therefore Maximum C_PDU Segment size is best set to 1023 bytes. If may be desirable to have smaller D_PDUs, which is best achieved by the multicast application choosing to use a smaller APDU size, which will constrain D_PDU size.

C.5.3. **Configuration of Maximum C_PDU Segment Size**

This annex requires that Maximum C_PDU Segment Size is configurable. It could be interpreted that this is a single fixed value, although notes in Edition 3 Annex H suggest otherwise. This section clarifies interpretation for use with both Edition 3 and Edition 4 peers.

For a given transmission, the Maximum C_PDU Segment Size (i.e., the largest C_PDU segment size used) used to generate new D_PDUs **shall not** vary. The value used **may** vary between transmissions, noting considerations on choice in Section C.5.1.1 and Section C.5.2.1.

At 75bps a D_PDU **may** take 110 seconds to transmit. Any constraints on maximum transmission time **shall not** lead to preventing transmission of a single D_PDU that needs a longer transmission time.

C.5.4. **Relaxation of Constraint on C_PDU Segment Size**

There is a constraint on handling segmentation and re-assembly that is relaxed in Edition 4. An implementation conforming to this specification **shall** handle this constraints on reception. An implementation **may** make use of this relaxation when sending to an Edition 4 (or subsequent) peer, but **shall not** use this constraint relaxation when sending to an Edition 3 peer.

When sending any D_PDU, the size **shall not** be greater than the Maximum C_PDU Segment Size but **may** be less than the Maximum C_PDU Segment Size. This relaxation of the Edition 3 constraints gives three benefits:

1. It allows space at the end of a transmission to be filled with a DATA-ONLY D_PDU, this giving better link utilization.
2. It allows a C_PDU to be split into segments of equal size, which will give better performance.
3. It allows D_PDUs to be aligned to waveform block boundaries. When there is fading, this often leads to blocks being badly corrupted. By ensuring that all D_PDUs are in a single modem block, performance is improved.

C.6. EOW and Management Message Types

The set of EOW Messages listed in this section **may** be placed in the 12-bit EOW section of any D_PDU. These messages are transmitted as information only, with no acknowledgement.

There is also a framework for acknowledged EOWs. This framework was used by EOWs in earlier version of this standard. Acknowledged EOWs are sent in the MANAGEMENT D_PDU.

The types of EOW used in STANAG 5066 are listed in Table C-4:

Table C-4. EOW Message Types

EOW Type	EOW Name	Description and Reference	Edition
0	EMPTY	Null EOW specified in Section C.6.1.	All
1	ED3-BASIC-RATE	Requesting basic rate, specified in Section C.6.2.	Ed3
2	ED3-BASIC-RATE-RESPONSE	Note 1	Ed3
3	UNRECOGNIZED-TYPE-ERROR	Error specified in Section C.6.3	All
4	ED3-CAPABILITY	Capability described in Section C.6.4	Ed3
5	ED3-ALM-REQUEST	Note 1	Ed3
6	ED3-ALM-RESPONSE	Note 1	Ed3
7	RESERVED		-
8	MAX-SPEED	Speed recommended for maximum throughput specified in Section C.6.5.	Ed4
9	LOW-SPEED	Speed recommended for low latency data specified in Section C.6.7	Ed4
10	MAX-SEGMENT	Recommended maximum C_PDU Segment Size specified in Section C.6.7.	Ed4
11	SENDER-APPROACH	Approach used by sender to guide receiver specified in Section C.6.8.	Ed4
12	SPEED-USED	Communicates speed used by sender specified in Section C.6.9.	Ed4
13	TOKEN	Sends Token in WTRP specified in Annex L.	Ed4
14	RESERVED		-
15	CAPABILITY	Capability described in Section C.6.10.	Ed4

Note 1. EOWs of types 2, 5 and 6 are only needed to support the optional procedures specified in Section C.7.6.1.2. If any of these procedures are implemented, the encoding of these EOWs **shall** be according to the specifications in Edition 3 Annex C.

The first column indicates the EOW number. The second column defines an EOW Name, that is used to refer to this EOW. The third column gives a brief description of the EOW and a reference to where the EOW is specified. The fourth column indicates the use based on peer support of STANAG 5066 Editions:

- "All" **may** be used with any peer.
- "Ed3" **may** be used with any peer known to support Edition 3 (or earlier). It **shall not** be used with any peer known to support Edition 4 (or later) or where peer support is unknown.
- Ed4" **may** be used with any peer known to support Edition 4 (or later). It **shall not** be used with any peer known to support Edition 3 (or earlier) or where peer support is unknown.

It is anticipated that support for the Ed3 EOWs will be dropped in a future edition of STANAG 5066.

Three EOWs are marked as RESERVED. These are EOWs used in previous editions of STANAG 5066. EOW 2 was used for a data rate change procedure for non-auto-baud waveforms that is not supported in this edition. EOW 7 was specified as an alternate rate change extended EOW, that is not widely used. EOWs 5 and 6 were used in Annex I, which has been removed from this edition. It is anticipated that these EOWs may be re-assigned in a future edition, if requirements for more EOWs arise.

The format of the EOW message types **shall** (3) be as shown in Figure C-46.

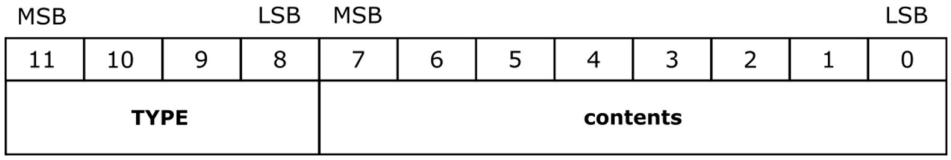


Figure C-45. Format of EOW Messages

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.6.1. EMPTY (TYPE 0) EOW Message

The EMPTY EOW (Type 0) has the contents set to all zero. This provides a default EOW to use when there are no other EOWs to communicate.

For Edition 4 peers, it is recommended to use the CAPABILITY EOW rather than EMPTY.

C.6.2. ED3-BASIC-RATE (TYPE 1) EOW Message

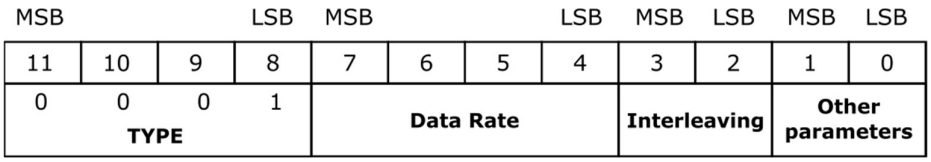


Figure C-46. Format of ED3-BASIC-RATE EOW Message.

The ED3-BASIC-RATE (TYPE 1) EOW Message **shall** (1) be used in conjunction with the Data Rate Selection Procedure, as specified in Section C.7.6.

The ED3-BASIC-RATE (TYPE 1) EOW Message **shall** be used by a receiving node as an advisory message indicating the modem parameters to which the link may be set to provide optimum performance.

The ED3-BASIC-RATE (TYPE 1) EOW Message **shall** (2) be formatted and encoded as specified in Figure C-46 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 recommends that the peer transmit data, in accordance with the encoding defined in the following table:

Table C-5. Data Rate Parameter ED3-BASIC-RATE

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	8000 bps
1 0 1 1	9600 bps
1 1 0 0	14400 bps
1 1 0 1	16000 bps
1 1 1 0	19200 bps
1 1 1 1	12800 2-ISB

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: ED3-BASIC-RATE

MSB - LSB	Interpretation
0 0	no interleaving
0 1	short
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 ED3-BASIC-RATE (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

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)

The encodings of Other Parameters is included for reference only. This field **shall** be set to "1 0" when transmitting this EOW and ignored when receiving it.

C.6.3. **UNRECOGNIZED-TYPE-ERROR (TYPE 3) EOW Message**

MSB 11	10	9	8	7	6	5	4	3	2	1	LSB 0
Type = 3				Reserved for Future Use				Unrecognized EOW Type			

Figure C-47. Format of UNRECOGNIZED-TYPE-ERROR EOW Message

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

A node **should** send an UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message to the originator of an unrecognised EOW message whenever the node receives an EOW that it does not recognize.

The UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message **shall** (2) be encoded as shown in Figure C-47 and include the following field:

- Unrecognized EOW Type

The type of the unrecognised EOW message or the message that triggered the error **shall** (3) be placed in the Unrecognized EOW Type.

The bits reserved for future use **shall** be set to the value zero (0).

C.6.4. **ED3-CAPABILITY (TYPE 4) EOW Message**

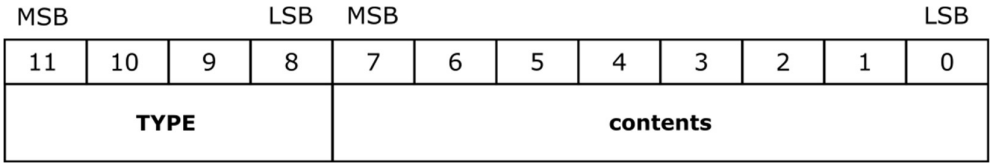


Figure C-48. Format of ED3-CAPABILITY EOW Message

The ED3-CAPABILITY Type 4 allows a node to inform another node of its capabilities related to modem parameters, waveform, and control.

This capability EOW is retained to enable capability exchange with an Edition 3 system. Capability exchange with an Edition 4 (or later) shall use the CAPABILITY (Type 15) EOW.

The ED3-CAPABILITY Type 4 **shall** (1) be encoded as shown in Figure C-48 and contains a single field, Contents.

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

Table C-8. Contents of Message field for ED3-CAPABILITY

bit	meaning
7 (MSB)	Adaptive modem parameters (DRC) capable note 1 (0 = no, 1 =
6	STANAG 4529 available note 2 (0 = no, 1 = yes)
5	MIL-STD-188-110 75-2400 bps 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 note 5 (0 = no, 1 = yes)
0 (LSB)	ARCS capable note 6 (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.
5. non-ARCS ALE capability may imply MIL-STD-188-141 Appendix A or proprietary capabilities and any ambiguity must be resolved outside of STANAG 5066.
6. ARCS –capable systems are those equipped with NATO's Automatic Radio Control System for HF Links, STANAG 4538 (FLSU/RLSU) or MIL-STD-188-141B (RLSU); any ambiguity must be resolved outside of STANAG 5066.

C.6.5. **MAX-SPEED (TYPE 8) EOW Message**

The MAX-SPEED (TYPE 8) EOW is sent by a node receiving data from a peer node to communicate to that node its recommendation for transmission speed to be used to obtain maximum throughput. It is likely that transmission at this speed will lead to a significant fraction of transmitted D_PDUs being corrupted and subsequently retransmitted.

The MAX-SPEED (TYPE 8) EOW is encoded as specified in Section C.6.5.1, which encodes a transmission speed and interleaver. The transmission speed is the recommended speed to achieve maximum throughput.

The Interleaver field specifies the minimum length Interleaver recommended for use with this transmission speed. Longer interleavers give better performance for data, and so it will often be appropriate for the sender to select a longer interleaver.

C.6.5.1. Transmission Speed and Interleaver Encoding

Several EOWs need a single byte representation of transmission speed and interleaver, so this is specified in an independent section. The encoding is shown in Figure C-49.

MSB 7	6	5	4	3	2	1	LSB 0
Interleaver			Speed				

Figure C-49. Transmission Speed and Interleaver Encoding.

Figure C-49 defines a one byte encoding of transmission speed and interleaver. Transmission speed is represented as a five bit Speed field and Interleaver as three bit Interleaver field. The values of these fields are specified in Figure C-50 and Figure C-51.

Value	Speed (bps)
0001-1101	WBHF. Waveform is the STANAG 5069 Waveform number (1-13)
10001	75
10010	150
10011	300
10100	600
10101	1200
10110	2400
10111	3200
11000	4800
11001	6400
11010	8000
11011	9600
11100	12800
11101	16000 2-ISB
11110	19200 2-ISB
Other	Reserved

Figure C-50. Speed Encoding of Transmission Speed/Interleaver

The Speed field specified transmission speed as shown in Figure C-50. This uses two ranges:

1. Values 00001-1101 (decimal 1-13) are used for Wideband HF (WBHF) waveforms as specified in STANAG 5069. STANAG 5069 defines 13 waveforms, with the waveform directly identified by the Speed value. Bandwidth (3kHz – 48 kHz) will be fixed for a sequence of transmissions, usually negotiated by 4G ALE. It is expected that this field will be generated and interpreted in context of the current bandwidth,
2. Values 10001-11100 are used for narrowband HF, and represent each of the standard speeds.

Value	Interleaver
000	No recommendation
001	Ultra Short
010	Very Short
011	Short
100	Medium
101	Long
110	Very Long
111	Reserved

Figure C-51. Interleaver Encoding of Transmission Speed/Interleaver

The Interleaver field is encoded as shown in Figure C-51, with the specified bits representing the chosen interleaver.

C.6.6. LOW-SPEED (TYPE 9) EOW Message

The LOW-SPEED (TYPE 9) EOW **may** be sent by a node receiving data from a peer node to communicate to that node its recommendation for maximum transmission speed to be used to used for data where low latency is desired. This is a speed where it is estimated by the receiver that only a very small fraction of D_PDUs will be corrupted on transmission.

The LOW-SPEED (TYPE 9) EOW is encoded as specified in Section C.6.5.1, which encodes a transmission speed and interleaver.

The recommended maximum transmission speed for low latency data is encoded directly. An interleaver is also encoded, which is the minimum length of interleaver recommended for low latency data.

C.6.7. MAX-SEGMENT (TYPE 10) EOW Message

The MAX-SEGMENT (TYPE 10) EOW **may** be sent by a node receiving data from a peer node to communicate to that node its recommendation for the maximum C_PDU segment size. This is particularly useful for fixed speed transmission, where reducing C_PDU segment size is the only option to deal with a poor quality link. It can also provide supplementary information for use in conjunction with MAX-SPEED EOW.

The MAX-SEGMENT (TYPE 10) EOW is encoded as an integer (range 0-255). The recommended maximum CPDU segment size is determined by multiplying this integer by four (4).

C.6.8. SENDER-APPROACH (TYPE 11) EOW Message

The SENDER-APPROACH (TYPE 11) EOW **may** be sent by a node transmitting data ta a peer node to communicate to that node information on its approach to sending data. This information can enable the receiving node to communicate back information that is most helpful to the sending node. The choices are discussed in Section C.7.6. This section defines the encoding.

MSB 7	6	5	4	3	2	1	LSB 0
Reserved for Future Use				EOW 12 Needed	Strategy		Fixed

Figure C-52. Format of SENDER-APPROACH EOW Message

The encoding of the SENDER-APPROACH (TYPE 11) EOW Message is shown Figure C-52.

- Fixed is bit 0. If Fixed is set to 1, transmission **shall** be at fixed speed. If Fixed is set to 0, transmission speed **may** be varied.
- Strategy (Bits 1 and 2) have the following meanings:
  - 00: Bulk data being transferred (to be optimized for throughput).
  - 01: Low latency data being transferred.
  - 10: A mix of bulk and low latency data being transferred.
  - 11: Bulk data with low latency requirements (e.g., Web browsing).
- If EOW 12 Needed (Bit 3) is set, this indicates that the local system cannot determine modem speed and/or interleaver of received transmissions. This **may** be treated by the receiver as a request to send SPEED-USED (TYPE 12) EOW Messages to the node. If EOW 12 Not Needed is not set the receiver should not send any SPEED-USED (TYPE 12) EOW Messages to the node.

- Bits 4-7 are reserved for future use and **shall** be set to 0.

It is anticipated that the basic use of this attribute will be service configuration. It is also possible that this value can be adaptive based on load and traffic being handled, perhaps by analysis of active SAPs.

C.6.9. SPEED-USED (TYPE 12) EOW Message

The SPEED-USED (TYPE 12) EOW **may** be sent by a node transmitting data to a peer node to communicate the transmission speed and interleaver being used in the current transmission. This is to provide this information when the receiver cannot determine this information locally.

The SPEED-USED (TYPE 12) EOW is encoded as specified in Section C.6.5.1, which encodes a transmission speed and interleaver.

C.6.10. CAPABILITY (TYPE 15) EOW Message

The CAPABILITY (TYPE 15) EOW is sent by a node to communicate to peers its capabilities.

MSB 7	6	5	4	3	2	1	LSB 0
Reserved for Future Use						Duplex	Ed4

Figure C-53. Format of CAPABILITY EOW Message

The format of CAPABILITY (TYPE 15) EOW is shown in Figure C-53.

- The Ed4 bit (bit 0) **shall** be set if the node supports Edition 4 of STANAG 5066 (or subsequent). Use of this information is summarized in Section C.3 “Support for Edition 3 Interoperability”.
- The Duplex bit (bit 1) is set if the node can support duplex capabilities with separate transmit and receive channels.
- Bits 2-7 are reserved for future use and **shall** be set to 0.

C.7. 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.7.1. 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-54 and as follows:

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

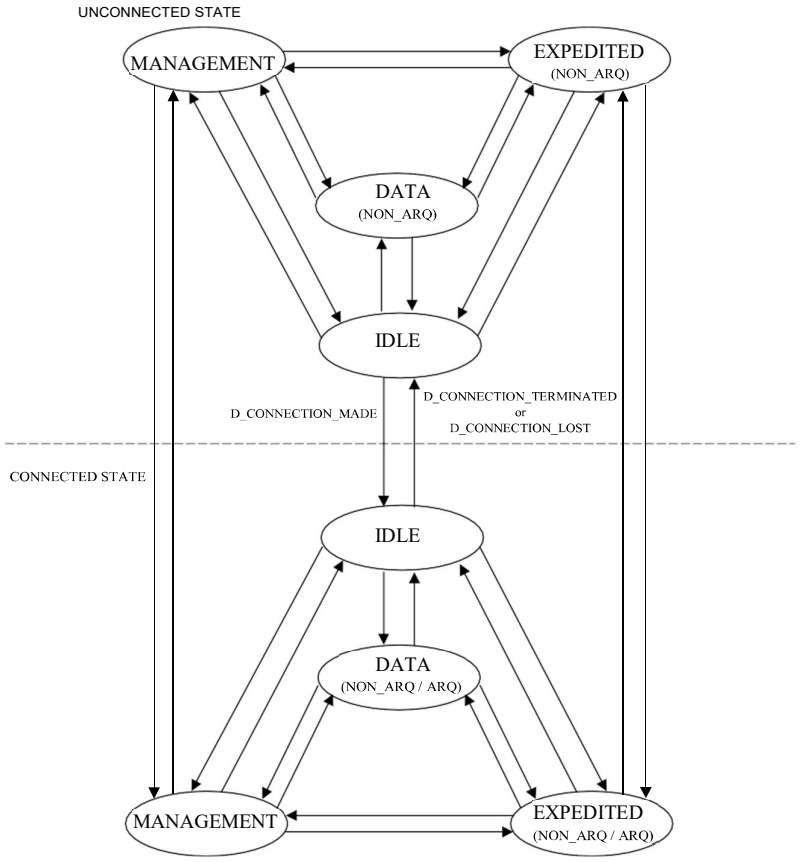


Figure C-54. Nominal State and Transition Diagram (Data Transfer Sublayer)

C.7.1.1. 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.7.1.1.1. 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-9. Event/Transition/Action Rules for IDLE (UNCONNECTED) State: CAS-Related Events

Received D_Primitive/	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.7.2 [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-10. Event/Transition/Action Rules for IDLE (UNCONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0/9 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
ACK Type 1/10 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
DATA-ACK Type 2/11 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
RESET/WIN RESYNC Type 3/12 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
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
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
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
EXTENSION Type 13 D_PDU (values 1-4)	IDLE(UNCONNECTED) , i.e., No Change.	Contents of D_PDU Handled by MAC Layer These values are specified in Annex L.
PADDING Type 14 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	Ignore
WARNING Type 15 D_PDU	IDLE(UNCONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.2. **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-11. Event/Transition/Action Rules for DATA (UNCONNECTED) State: CAS-Related Events

Received D_Primitive	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.7.2 [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-12. Event/Transition/Action Rules for DATA (UNCONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 /11 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 3/12 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
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_
EXTENSION Type 13 D_PDU (values 1-4)	DATA(UNCONNECTED), i.e., No Change.	Contents of D_PDU Handled by MAC Layer. These values are specified in Annex L.
PADDING Type 14 D_PDU	DATA(UNCONNECTED), i.e., No Change.	Ignore
WARNING Type 15 D_PDU	DATA(UNCONNECTED), i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.3. **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-13. Event/Transition/Action Rules for EXPEDITED-DATA (UNCONNECTED) State: CAS-Related Event

Received D_Primitive	State Transition to:	Action & Comments
D_CONNECTION_MADE	IDLE(CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.7.2 [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.,	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-14. Event/Transition/Action Rules for EXPEDITED-DATA (UNCONNECTED) State: Reception-Related Event

Received D_PDU/ Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 /11D_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 3/12 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
EXTENSION Type 13 D_PDU (values 1-4)	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	Contents of D_PDU Handled by MAC Layer
PADDING Type 14 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	Ignore
WARNING Type 15 D_PDU	EXPEDITED-DATA (UNCONNECTED), i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.4. 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-15. Event/Transition/Action Rules for MANAGEMENT (UNCONNECTED) State: CAS-Related Events

Received D_Primitive/	State Transition to:	Action & Comments
D_CONNECTION_MADE	MANAGEMENT (CONNECTED)	INITIALIZE State Variables for ARQ processing, in accordance with Section C.7.2 [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	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-16. Event/Transition/Action Rules for MANAGEMENT (UNCONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 /11D_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 3/12 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
EXTENSION Type 13 D_PDU (values 1-4)	MANAGEMENT (UNCONNECTED) , i.e., No Change.	Contents of D_PDU Handled by MAC Layer
PADDING Type 14 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	Ignore

WARNING Type 15 D_PDU	MANAGEMENT (UNCONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.5. **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-17. Event/Transition/Action Rules for IDLE (CONNECTED) State: CAS-Related Event

Received D_Primitive	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-18. Event/Transition/Action Rules for IDLE (CONNECTED) State: Reception-Related Event

Received D_PDU/ Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 2/11 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 3/12 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

EXTENSION Type 13 D_PDU (values 1-4)	IDLE(CONNECTED), i.e. No Change	Contents of D_PDU Handled by MAC Layer. These values are specified in Annex L.
PADDING Type 14 D_PDU	IDLE(CONNECTED), i.e. No Change	Ignore
WARNING Type 15 D_PDU	IDLE(CONNECTED), i.e. No Change	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.6. **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-19. Event/Transition/Action Rules for DATA (CONNECTED) State: CAS-Related Events

Received D_Primitive	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-20. Event/Transition/Action Rules for DATA (CONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 /11D_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 3/12 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
EXTENSION Type 13 D_PDU (values 1-4)	DATA(CONNECTED), i.e. No Change	Contents of D_PDU Handled by MAC Layer. These values are specified in Annex L.
PADDING Type 14 D_PDU	DATA(CONNECTED), i.e. No Change	Ignore
WARNING Type 15 D_PDU	DATA(CONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.7. **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-21. Event/Transition/Action Rules for EXPEDITED-DATA (CONNECTED) State: CAS-Related Events

Received D_Primitive	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.7.1.3: IF completed transmitting all Expedited-Data, change to DATA(CONNECTED), OTHERWISE, EXPEDITED-DATA (CONNECTED), i.e., No Change.	in Accordance with Section C.7.1.3: 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.7.1.3: EXPEDITED-DATA (CONNECTED), i.e. No Change	in Accordance with Section C.7.1.3: 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.7.1.3: IF completed transmitting all Expedited-Data, change to DATA(CONNECTED), OTHERWISE, EXPEDITED-DATA (CONNECTED), i.e., No Change.	in Accordance with Section C.7.1.3: 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-22. Event/Transition/Action Rules for EXPEDITED-DATA (CONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 /11D_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 3/12 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
EXTENSION Type 13 D_PDU (values 1-4)	EXPEDITED-DATA (CONNECTED), i.e. No Change	Contents of D_PDU Handled by MAC Layer. These values are specified in Annex L.
PADDING Type 14 D_PDU	EXPEDITED-DATA (CONNECTED), i.e. No Change	Ignore

WARNING Type 15 D_PDU	EXPEDITED-DATA (CONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.1.8. **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-23. Event/Transition/Action Rules for MANAGEMENT (CONNECTED) State: CAS-Related Events

Received D_Primitive/	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	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

	Change.	management protocol.
D_EXPEDITED_UNIDATA_ REQUEST	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-24. Event/Transition/Action Rules for MANAGEMENT (CONNECTED) State: Reception-Related Events

Received D_PDU Event	State Transition to:	Action & Comments
DATA Type 0/9 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 1/10 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 /11D_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 3/12 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
EXTENSION Type 13 D_PDU (values 1-4)	MANAGEMENT (CONNECTED) , i.e., No Change.	Contents of D_PDU Handled by MAC Layer. These values are specified in Annex L.
PADDING Type 14 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	Ignore
WARNING Type 15 D_PDU	MANAGEMENT (CONNECTED) , i.e., No Change.	CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED

C.7.1.2. 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 13, 14 or 15 D_PDUs
DATA(UNCONNECTED)	type 7, 13, 14 or 15 D_PDUs
EXPEDITED DATA(UNCONNECTED)	type 8, 13, 14 or 15 D_PDUs
MANAGEMENT(UNCONNECTED)	type 6, 13, 14 or 15 D_PDUs
IDLE(CONNECTED)	type 13, 14 15 D_PDUs
DATA(CONNECTED)	type 0, 1, 2, 3, 7, 8, 9, 10, 11, 12, 13, 14 or 15
EXPEDITED DATA(CONNECTED)	type 1, 4, 5, 8, 10, 13, 14 or 15 D_PDUs
MANAGEMENT(CONNECTED)	type 6, 8, 13, 14 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 C.7.1.1.

C.7.1.3. 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 0/9 and 2/11.

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

Prior to reaching DATA STATE, CAS-1 negotiation will have determined whether to use the family of D_PDUs with 16 bit frame sequence number (Types 9, 10, 11 and 12) or the family of D_PDUs with 8 bit frame sequence number (Types 0, 1, 2 and 3). The logic of DATA state processing is the same for each family, with the different frame sequence number size leading to a different modulo (256 for 8 bit and 65,536 for 16 bit). The logic is described using the generic D_PDU names, which is also the same as the 16 bit frame sequence number D_PDU names.

Examples in this section use both numbers. Note that the two families of D_PDU **shall not** be mixed.

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-55 and Figure C-56.

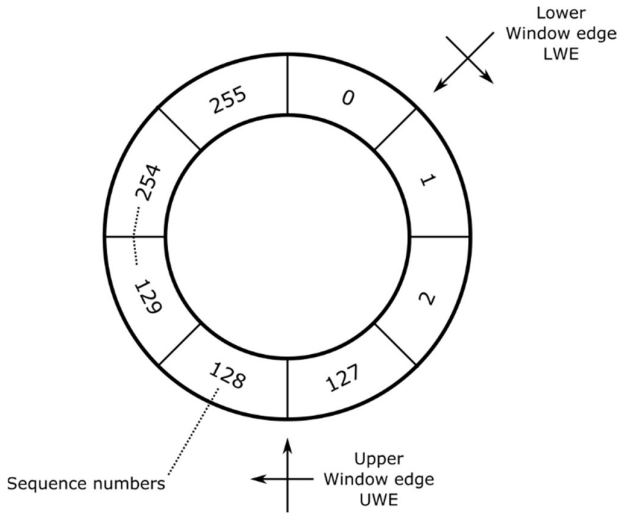


Figure C-55. Sequence-number Space (8 bit): Integers 0..255

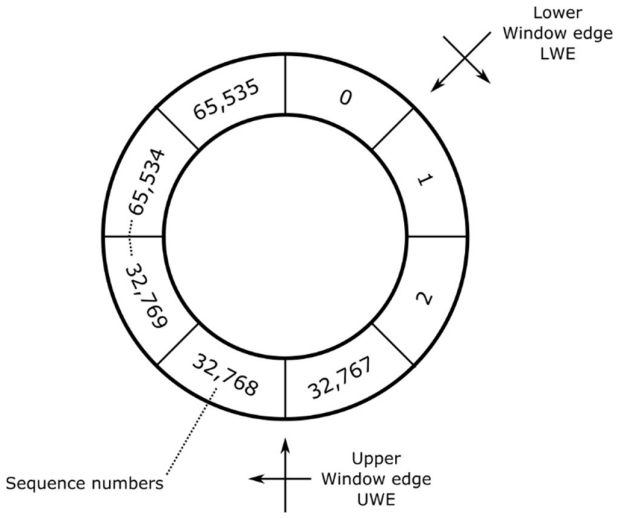


Figure C-56. Sequence-number Space (16 bit): Integers 0..65,535

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 or 65,536 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 or 65,536 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 or 65,536 operations).

f) The difference (as a modulo-256 or 65,536 arithmetic operator) between the TX UWE and TX LWE – 1 **shall** be equal to the “current transmitter window size”.

g) The “maximum window size” **shall** (8) equal 128 or 32,768.

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 or 32,768 (“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 or 65,536) 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 or 65,536 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 or 65,536) from the RX LWE. The RX UWE increases monotonically as D_PDUs with higher (mod 256 or 65,536) 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 or 32,768 **shall** (14) not be accepted or acknowledged by the receiver node.

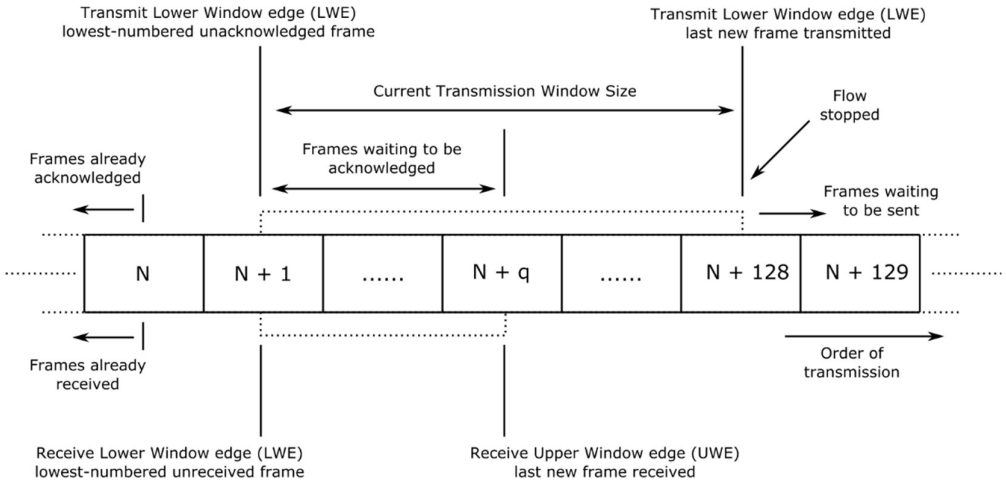
o) If the value of the RX LWE field in any ACK Type 1 or DATA-ACK Type 2 D_PDU is greater than the TX LWE, the Data Transfer Sublayer **shall** (16) declare all D_PDUs with Frame Sequence Numbers between the TX LWE and the RX LWE value as acknowledged, and **shall** (17) advance the TX LWE by setting 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 or 65,535
- RX LWE = 0
- RX UWE = 255 or 65,535

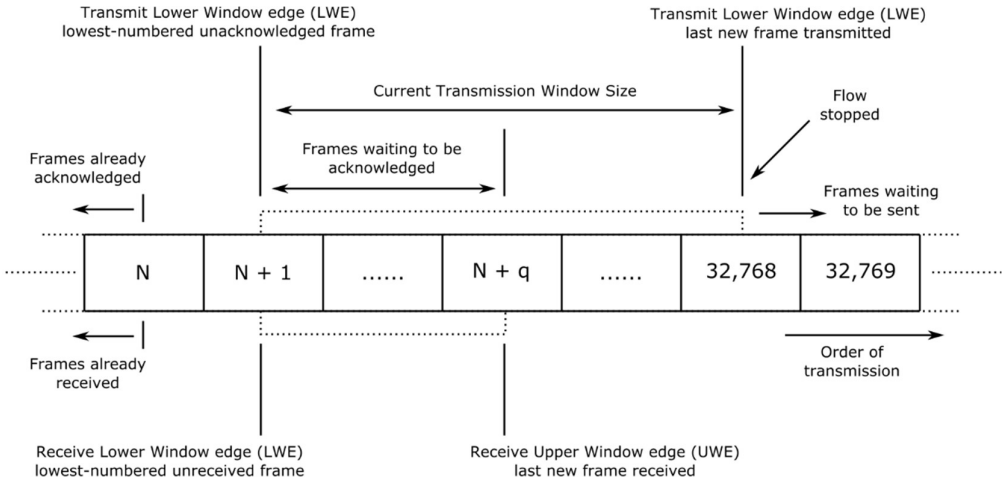
[Note that, although the flow control limitations limit the number of D_PDUs to be transmitted in one transmission, other protocol parameters will also effect this in an indirect way; the structure of the EOT parameter allows a maximum transmission interval of about 2 minutes. If a D_PDU size of 200 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. At higher speeds, the 128 D_PDU limit of 8bit frame sequence numbers impacts performance, and so use of 16 bit frame sequence number is desirable.]

The nominal relationships between the ARQ Flow-Control variables specified above are shown in Figure C-57 Figure C-58.



ARQ Flow-Control Variables at the Receiver

Figure C-57. ARQ Flow Control Window Variables (8 bit)



ARQ Flow-Control Variables at the Receiver

Figure C-58. ARQ Flow Control Window Variables (16 bit)

C.7.3. Synchronisation of the ARQ Machine

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

C.7.3.1. 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 Section C.7.2).

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.7.3.2. 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]

Figure C-59 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.

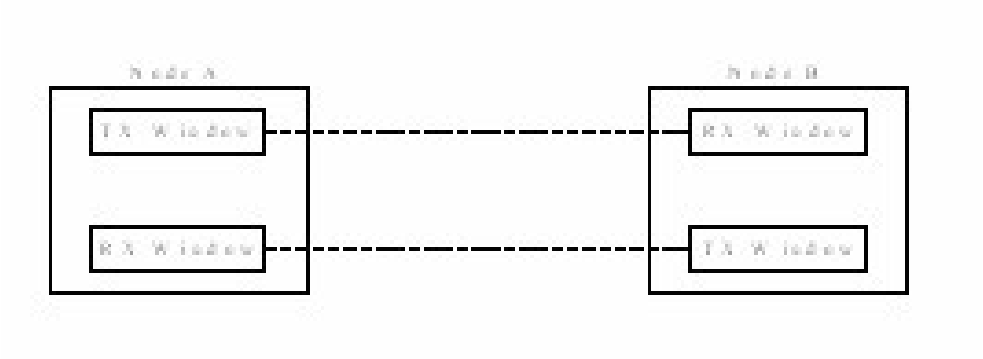


Figure C-59. Synchronization Points

In the event that it is determined that the nodes are not synchronized, the nodes **shall** be synchronized following the procedure set out in Section C.7.4.

C.7.3.2.1. **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 or 32,768 and therefore cannot be greater than the

{RX LWE + MAX WIN SIZE –1} (modulo 256 or 65,536). This condition is expressed by the second part of equation 1. Equation 1 can therefore be used to establish whether the TX UWE is *in synchronisation* with the RX window edges and *loss of synchronisation* has occurred if this equation is not satisfied.

Note 1: 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-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.

Note 2: All the synchronisation tests assume a fixed window size equivalent to the maximum window size of 128 or 32,768 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

The *in synchronisation* and *out of synchronisation* regions of the FSN circle described by equation 1 are illustrated graphically in Figure C-60.

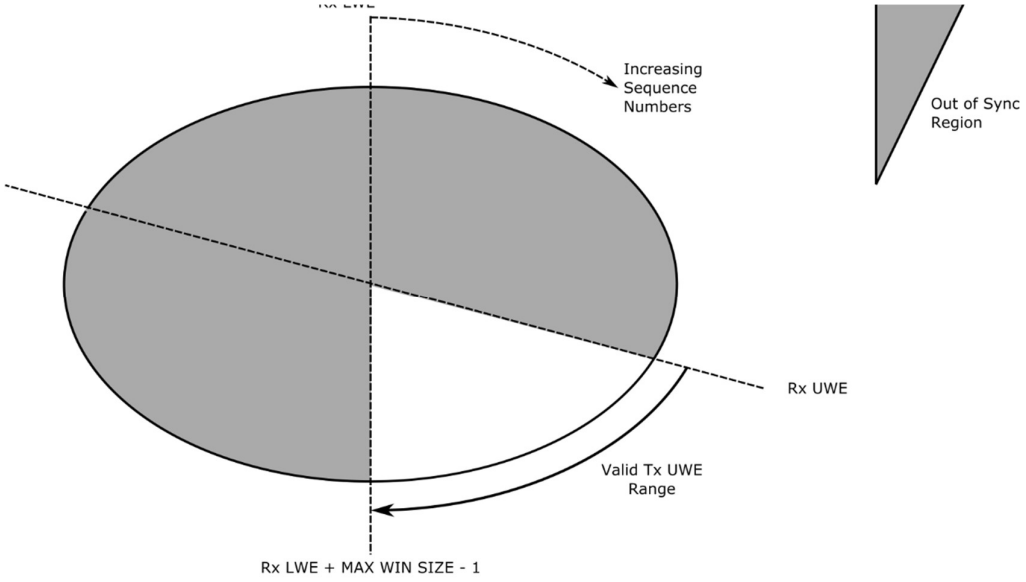


Figure C-60. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 1)

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-61.

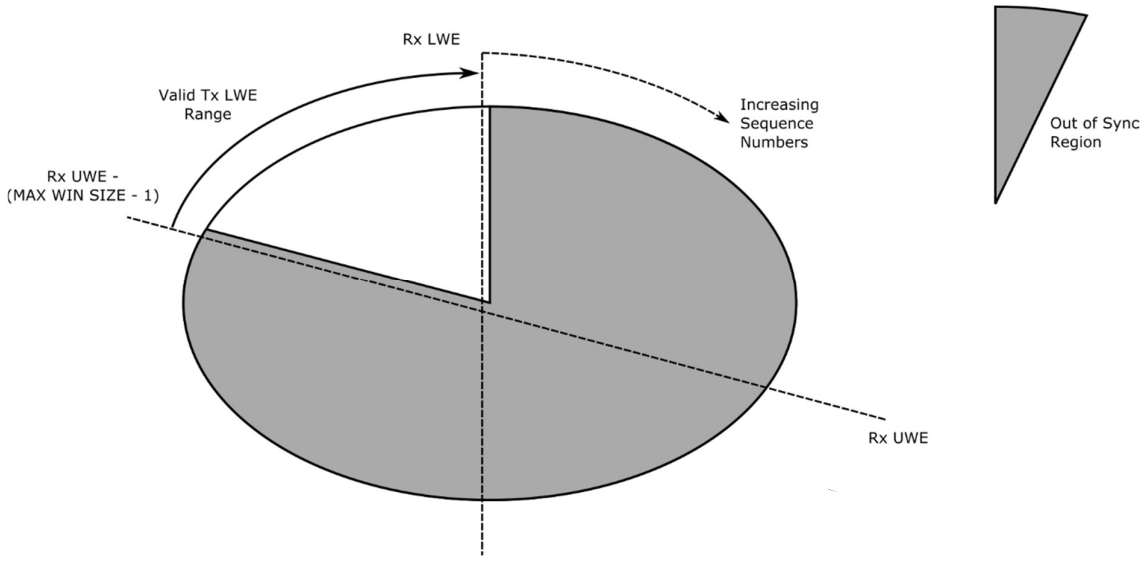


Figure C-61. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 2)

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-62.

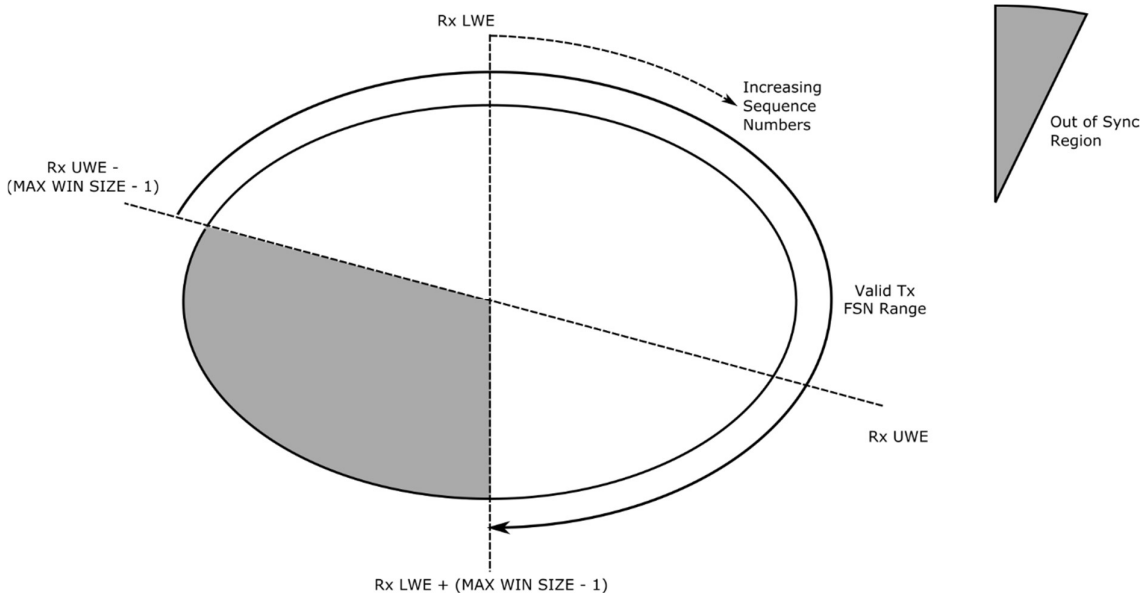


Figure C-62. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 3)

C.7.3.2.2. 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-63.

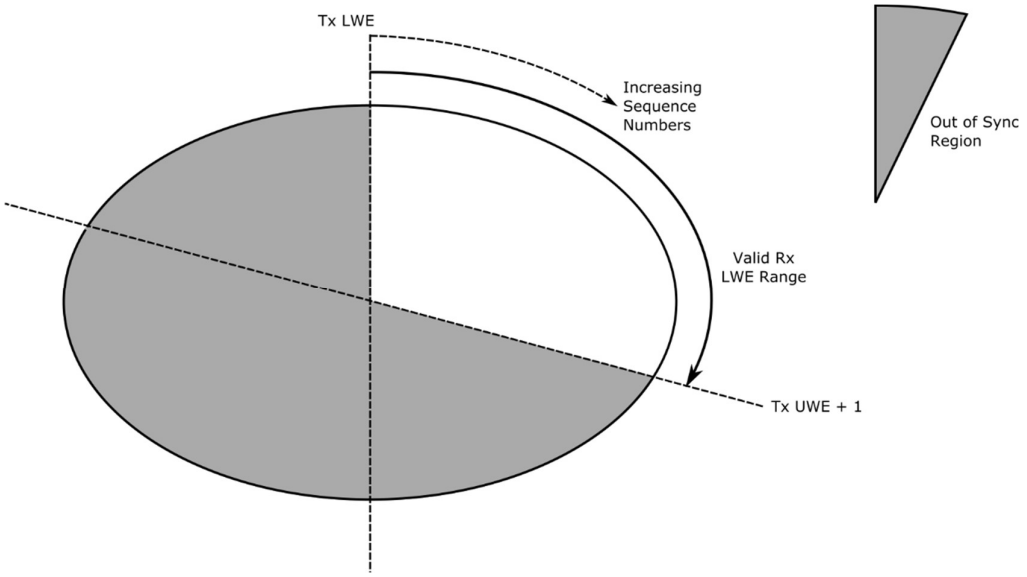


Figure C-63. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 4)

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-64.

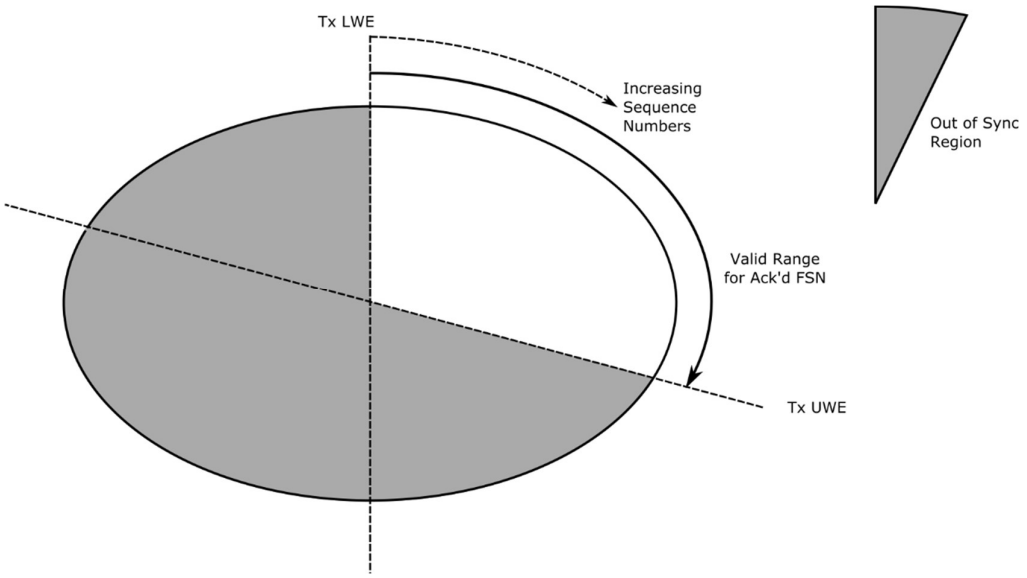


Figure C-64. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 5)

C.7.4. Procedure for use of Type 3 RESET/WIN RESYNC D_PDU

The Type 3 RESET/WIN RESYNC D_PDU, specified in Section C.4.7, 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.7.2);
- 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.

C.7.5. **Transmitting and Receiving D_PDUs**

The DTS state machine and procedures for operation define operation in terms of D_PDUs sent and received. This section describes how this state machine interacts with the channel and how transmissions are structured. It considers three types of configuration:

1. Single Channel, which will send and receive D_PDUs.
2. Duplex Channel Pair.
3. Broadcast Channel.

C.7.5.1. **Receiving D_PDUs**

Incoming D_PDUs are handled as they are read from the channel. If a D_PDU header is parsed and the header checksum is correct, the following actions are taken:

1. If WTRP (Annex L) is being used:
    a. Pass EOW 13 or 14 to the WTRP layer.
    b. Pass sender and receiver information to the WTRP layer, in support of WTRP promiscuous mode, along with any information on receive signal SNR and quality.
2. Pass any EOW, apart from EOW 13 or 14 to the state machine.
3. Handle EOTs as described below.
4. Make available any information on receive signal SNR and quality to the subsystem described in Section C.7.6.

After this initial processing, validate the Data Checksum for D_PDUs with such a checksum. If this validation fails, make DATA-ONLY and DATA-ACK D_PDUs to the state machine for processing the Non-ARQ with Errors service. Otherwise discard the D_PDU.

Pass Extended D_PDUs of types 1-6 to the WTRP layer. Discard Extended D_PDUs of any other type. All other D_PDUs are passed to the state machine.

C.7.5.1.1. **Switching To Transmission**

A node operating in single channel mode **may** transmit after receiving a transmission. The rules for when a node may transmit are set out in Annex K (CSMA) and Annex L (WTRP). One or other of these procedures **shall** be followed.

In addition to this, the receiving system **shall** process received EOTs. Any EOT received allows the receiving system to calculate when the transmission will end. Information from multiple EOTs **shall** be used to determine this end point time as accurately as possible. The node **shall not** transmit before this end point time is reached, even if no signal is being received from the channel.

When a transmission is being received and no EOTs are parsed, it is also desirable to pass modem information on the end of transmission to the MAC layer, to help with transmission control.

C.7.5.2. **D_PDU Queue**

On transmission side, the state machine will lead to a sequence of D_PDUs being generated for transmission. These D_PDUs can be considered to be in a queue. This queue is specified to enable description of the approach to be taken. An implementation is not required to work in this way, provided that the external effect resulting is the same as one from the approach specified here.

Once a D_PDU has been transmitted, it is removed from the queue, with the exception of Non-ARQ-DATA D_PDUs, which **shall** be retained in the queue until the required

number of transmissions has been made.

C.7.5.2.1. Transmission Order Considerations

There is no requirement to transmit D_PDUs from the queue in the order that they arrive. A number of considerations for this choice are noted in the following sections. Two general considerations **shall** apply in all circumstances:

1. When an ARQ C_PDU is segmented, the D_PDUs with the C_PDU fragments **shall** be assigned adjacent frame sequence numbers and initial transmission **shall** be in frame sequence number order without insertion of other D_PDUs, other than retransmissions.
2. Where D_PDUs have differing priority, the first transmission of D_PDUs of higher priority **shall** always be made before the first transmission of D_PDUs of lower priority, when this does not conflict with the previous requirement or with “in order” handling
3. When it is determined from acknowledgements that transfer of a DATA-ONLY D_PDU has failed, retransmission of this D_PDU **shall** be given priority over initial transmission of data D_PDUs of the same priority.
4. Where there are no other factors, transmission of D_PDUs **shall** be “oldest first”.

Note that the DTS will handle D_PDUs for multiple peers. These rules apply across all peers, as the DTS can transmit data to any valid current destination.

C.7.5.2.2. Handling TTL Expiry

While D_PDUs are being transmitted, the TTL of one or more C_PDUs in transit **may** expire. There are various reasons this may happen, including handling of higher priority data or delays in transmission due to poor channel conditions. C_PDUs with TTL expired **shall not** be transmitted. One of the following two processes **shall** be followed for ARQ data in order to drop the D_PDUs while maintaining synchronization of the window.

When a C_PDU is dropped due to TTL, it **shall** be rejected to the CAS using either D_UNIDATA_REQUEST_REJECTED or D_EXPEDITED_UNIDATA_REQUEST_REJECTED with reason TTL expired.

A C_PDU **may** be dropped by use of the Drop PDU on each segment, as specified in Section C.4.4. No data is transmitted with this D_PDU, as it is simply used to transfer the segment. The dropped D_PDU **shall** be acknowledged. All D_PDUS comprising segments of the dropped C_PDU **shall** be transmitted and acknowledged.

Alternatively, the window **may** be reset using the procedure described in Section C.7.4. This approach may be more efficient if there are a large number of whole or partial C_PDUs to be dropped.

For non-ARQ data, and queued D_PDUs from C_PDUs with expired TTL **shall** be

dropped and not transmitted.

C.7.5.2.1. Handling ARQ Link Termination

An ARQ link **may** be terminated either because D_CONNECTION_TERMINATED received from the CAS or due to timeouts or other DTS failures. When this happens, C_PDUs that have not been fully processed **shall** be rejected to the CAS using either D_UNIDATA_REQUEST_REJECTED or D_EXPEDITED_UNIDATA_REQUEST_REJECTED with the reason taken from D_CONNECTION_TERMINATED or the local reason.

C.7.5.2.1. Handling D_PDUs Not Addressed to Local Node

The DTS state machine handles D_PDUs addressed to the local node by unicast or multicast address.

Although other D_PDUs are not handled by the state machine, implementation experience suggests that it is useful to monitor them. The choice and mechanisms to achieve this are an implementation decision.

C.7.5.3. Transmission Length and Parameters - General Considerations

There are a number of parameters that need to be chosen at start of transmission:

1. Transmission Length.
2. Transmission Speed.
3. Interleaver
4. Maximum C_PDU Segment Size.

General guidance and approach on selecting 2-4 is given in Section C.7.6. Selection specific to the various models are given below.

C.7.5.3.1. Modem Block Alignment

This section defines a procedure only available for Edition 4 (or subsequent) peers. It **shall not** be used when transmitting to an Edition 3 peer. It **may** be used for Edition 4 peers.

When severe errors occur on a HF link, they can lead to corruption of a modem data block where no D_PDUs transmitted in that block are received. Because of this, it is desirable to avoid splitting D_PDUs between modem blocks. DATA-ONLY D_PDUs will often contain C_PDU fragments. The C_PDPU segment size can be adjusted so that D_PDU and modem block boundaries are aligned. Note that there are anticipated to be non-trivial implementation issues to make this work as described.

When there are no alignment issues, the optimum choice for C_PDU segment size is one that is equal for all the C_PDU fragments of a given C_PDU, and as large as possible while being less that the maximum C_PDU segment size.

C.7.5.4. Single Channel

In the single channel model, nodes transmit in turn with rules for transmission governed by the chosen MAC layer (WTRP or CSMA). Each node will make a transmission of up to 127.5 seconds containing some or all of the queued D_PDUs.

C.7.5.4.1. Transmission Length Choice

For single channel, all of the parameters set out in Section C.7.5.3 need to be set at the start of transmission. In particular, the length of transmission needs to be fixed. The choice of transmission length is an implementation choice. This section notes a number of considerations in this choice:

1. The maximum transmission length allowed is 127.5 seconds.
2. Use of longer transmissions will optimize throughput.
3. At lower speeds, use of the longest transmission time allowed is generally desirable.
4. At higher speeds is may be desirable to use shorter maximum transmissions. This will reduce throughput and improve latency. This choice may be sensible where a mixture of applications with different QoS requirements are being supported.
5. Note that transmission of a maximum length DATA-ONLY D_PDU at 75 bps takes around 110 seconds, Any constraints on maximum transmission time **shall not** be implemented in a manner that can cause transfer or large D_PDUs at low speeds to be blocked.
6. Interleaver choice is discussed in Section C.7.6. Minimum transmission length will be a single interleaver block.
7. When there is queued traffic of different priorities, a node **may** choose to send only the higher priority traffic, leaving lower priority traffic for a subsequent transmission.

Note that although the length of transmission needs to be fixed at start of transmission, the transmission content does not. It is **recommended** to choose which D_PDUs to transmit as late as possible, so that higher priority D_PDUs or D_PDUs from applications with low latency QoS requirements that arrive after the start of transmission can be included in the transmission. Note that there are currently no mechanisms specified in STANAG to enable this QoS to be determined.

C.7.5.4.2. Repeating and Distributing D_PDUs

It is often beneficial to repeat D_PDUs. In particular:

1. It is generally desirable to repeat ACK-ONLY D_PDUs several times. These are small D_PDUs, and loss of Acks will lead to delays and the overhead of repeat transmission. Repeating transmission will minimize loss.
2. DATA-ONLY D_PDUs from applications which have low latency QoS, particularly when a long transmission is being used with aggressive speed choice optimizing for throughput. Repeating these D_PDUs will improve latency for the application concerned.
3. Non-ARQ D_PDUs with request for repeat transmissions. These repeats **may** be made within a single transmission or over multiple transmissions.

When a D_PDU is repeated, it is desirable to spread the repeated D_PDUs over the transmission. This separation will help to avoid long fades in the channel.

Transmissions will be an exact number of modem block lengths, and not an arbitrary size. This can lead to a situation where there is “extra space” to be used in a transmission. To optimize performance it is desirable that this space be filled with D_PDUs to the maximum extent possible. Two choices to do this are noted:

1. ACK-ONLY D_PDUs are generally small and useful to repeat. In many situations, they give a practical way to fill space.
2. PADDING D_PDUs are very small. They provide a way to fill space and repeat EOT and EOW transmission.
3. (Edition 4 only) When C_PDUs are being fragmented into multiple D_PDUs, the C_PDU segment size can be carefully chosen to ensure that blocks are filled.

C.7.5.4.3. **EOT Handling**

For single channel operations the EOT field **shall** be set in every D_PDU.

A node **shall not** make a transmission when the EOTs from transmission by another node indicate that it is transmitting.

Once an EOT is sent, the EOT in each subsequent D_PDU in that transmission **shall** 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 node **shall** stop transmitting when the EOT becomes zero. Rules for subsequent behavior are governed by the MAC layer, with CSMA procedures specified in Annex K and WTRP procedures specified in Annex L.

C.7.5.4.4. Transmitting to Multiple Peers

A node transmitting on a single channel **may** address all traffic to a single peer node. It **may** also address ARQ traffic to multiple nodes, with a CAS-1 soft link maintained with each node. It **may** also send non-ARQ traffic to unicast or broadcast addresses. Peers will take it in turns to transmit, with CSMA procedures specified in Annex K and WTRP procedures specified in Annex L.

In determining transmission parameters for a transmission being received by multiple, the needs of each of these peers **shall** be taken into consideration, which at a minimum **shall** be to not exceed the maximum recommended speed for each peer.

C.7.5.5. Duplex

A peer can be configured with separate transmit and receive channels. When this is done, the CAS will ensure that all traffic to the DTS is for that single peer. So when operating over duplex, the DTS will only be handling D_PDUs to be exchanged with a single peer.

C.7.5.5.1. Transmission Gaps vs Continuous

When a duplex channel has no current data to transmit, two strategies are available:

1. “Gaps”. When there is no more data to transmit, terminate the channel transmission. This has the advantage of enabling new parameters to be selected for the next transmission, but there is a delay in establishing the new transmission.
2. “Continuous”. Where the transmission is continued, using either ACK-ONLY D_PDUs noting the current window position or PADDING D_PDUs. This allows the channel to be more responsive to new D_PDUs arriving.

On reception, both modes **shall** be supported. For transmission, an implementation can choose to support one or both strategies.

C.7.5.5.2. Transmission Length and Parameters

With a duplex channel, the length of transmission does not need to be decided in advance and transmission can be of arbitrary length.

Transmission speed and interleaver need to be fixed at start of each transmission, using considerations discussed in Section C.7.6. If conditions and considerations change, a transmission can be terminated and a new transmission started with different parameters.

Sometimes channels will have low volumes of data. For example:

1. XMPP Chat communication with short messages.

2. A reverse channel to one handling a bulk data stream that is just passing back acknowledgements.

For traffic of this nature, it is desirable to choose a conservative transmission that minimizes data loss.

C.7.5.5.3. **Acknowledgement Handling**

When operating over a duplex channel, acknowledgements can be sent immediately. This contrasts to single channel, where acknowledgements are only sent after a (potentially long) transmission by the peer.

At lower speeds, where transmission of DATA-ONLY D_PDUs can take several seconds or more, it makes sense to send an ACK-ONLY D_PDU for each DATA-ONLY D_PDU received, and it **may** be beneficial to repeat transmission of each ack. At higher speeds, with many DATA-ONLY D_PDUs arriving each second, it **may** be preferable to acknowledge several DATA-ONLY D_PDUs with a single ACK-ONLY D_PDU.

When data is flowing in both directions, it will generally be desirable to perform acknowledgement with DATA-ACK D_PDUs. Repeat acknowledgements should use ACK-ONLY D_PDUs.

C.7.5.5.4. **EOT Handling**

In duplex mode, the length of transmission is generally not known. When this is the case, the EOT field **shall** be filled with all zeros to communicate that no EOT is set. If the user of a duplex channel determines when a transmission will end, it **may** continue to set EOT to all zeros or it **may** use EOT to record time remaining following the rules of Section C.7.5.4.3.

C.7.5.6. **D_PDU Retransmission**

When ARQ data is transmitted, D_PDUs that have failed to be transmitted **shall** be retransmitted. Retransmission of a failed D_PDU **shall** be given priority over initial transmission of a D_PDU of the same or lower priority. There are situations where it is possible but not certain that transmission of a D_PDU has failed. In these situations the potentially failed D_PDU **may** be retransmitted.

When an ACK-ONLY or DATA-ACK D_PDU is received, the LWE is advanced to indicate the reception point. The DATA-ACK D_PDU may also indicate specific D_PDUs numbered above the LWE that have been received and by implication a set of D_PDUs that have not been received. Note that multiple ACK-ONLY D_PDUs may be used, each of which will cover a specific segment of the ring space above the LWE, and that these ACK-ONLYs only indicate “not received” for the ring space that

is covered. D_PDUs so identified **shall** be considered to have failed to be transmitted.

The node will know the D_PDU with furthest advanced FSN that has been transmitted. This can be used to identify zero or more D_PDUs after the last explicitly acknowledged D_DPDU that **may** have been received. For duplex transmission, these D_PDU **shall not** be considered to have failed transmission. For half duplex it is **recommended** that these D_PDUs are considered to have failed transmission, taking into account that a node **may** start to transmit (at STANAG 5066 level) before an incoming transmission has been fully received in order to optimize turnaround time.

There are additional considerations when the MAC layer (CSMA or WTRP) indicates to a node that it can transmit, but no ack has been received to previously transmitted D_PDUs. There are three possibilities:

1. The transmitted D_PDUs were lost, and so no ack was sent; or
2. The ack was lost in transmission; or
3. The node which would have sent the ack transferred (higher priority) D_PDUs to another node and did not transmit the ack.

It is not in general possible to distinguish between these cases. Two strategies are possible:

1. Do not retransmit D_PDUs previously sent. This is the best option in cases 2 and 3.
2. Retransmit the D_PDUs previously sent. This is the best option in case 1.

An node **may** choose to use either strategy.

Some hints may be obtained from information that has been received, that can guide the choice:

1. If a transmission is received, which is likely or certain to have come from the CAS-1 peer, but no D_PDUs are parsed, case 1 is likely.
2. If WTRP is being used, it can be determined if the CAS-1 peer has transmitted. If it has dropped out of the ring, the CAS-1 link **shall** be terminated. If all D_PDUs were parsed (no gaps in transmission), case 2 can be eliminated and the choice between cases 1 and 3 inferred from the traffic received.

The QoS of traffic being handled can also guide the choice:

1. For higher priority traffic, it is more desirable to retransmit.
2. For low latency traffic, such as XMPP chat, it is more desirable to retransmit.
3. For bulk traffic, not transmitting is generally going to give best performance.

There are currently no mechanisms specified in STANAG 5066 to explicitly distinguish low latency and bulk, although low latency requirements can sometimes be inferred

by small amounts of data being sent.

C.7.5.7. D_PDU Broadcast

A node operating in broadcast mode will send (non-ARQ) data over the channel to multiple nodes and will never receive data back over the channel.

Handling a broadcast channel is similar to a duplex channel and the following rules apply:

1. A broadcast channel has transmission choices of “continuous” and “gaps” described in Section C.7.5.5.1.
2. Transmission length selection follows Section C.7.5.5.2.
3. EOT handling follows Section C.7.5.5.4.

C.7.6. Data Rate Selection

This section describes selection of transmission speed and related parameters.

C.7.6.1. Use of Edition 3 Data Rate Change for Non-Autobaud Waveforms

This specification supports data rate selection only for autobaud waveforms such as STANAG 4539 and STANAG 5069.

STANAG 5066 Edition 3 specifies a mechanism to perform stop/start Data Rate Change with non-autobaud waveforms. This mechanism **shall not** be used with Edition 4 peers, except as specified in Section C.7.6.1.3 below.

C.7.6.1.1. Minimum Edition 3 Interoperability

The stop/start data rate change procedure and the associated frequency change procedures make use of Management D_PDUs (type 6) with Type 1 and Type 2 EOWs contained. The frequency change of Annex I uses Management D_PDUs (type 6) with types 5 and 6 EOWs.

If one of these PDUs is received from an Edition 3 peer, the node **shall** send a Warning D_PDU (type 15) to indicate that the procedure is not supported, unless it follows the procedures of Section C.7.6.1.2.

C.7.6.1.2. Full Edition 3 Interoperability including Frequency Change

An implementation of the current version of Annex C **may** support the stop start Data Rate Change mechanism specified in Section C.6.4 of Edition 3 for interoperability with Edition 3 systems only.

An implementation **may** also support the Frequency Change mechanism specified in Annex I of Edition 3 for interoperability with Edition 3 systems only.

Note that these procedures only work with a single active peer, and so cannot be used if there are multiple active physical links.

C.7.6.1.3. **Interoperability with Edition 4 nodes using variable speed over a non-autobaud network**

In a mixed Edition 3 and Edition 4 network, interoperability between Edition 4 nodes will follow Edition 4 protocol in most situations. However, Edition 4 variable speed requires use of an autobaud waveform. In order to operate with an Edition 4 peer with variable speed over a non-autobaud network, a node **may** operate following Edition 3 procedures and achieve this by not advertising Edition 4 capability.

C.7.6.2. **Model for Data Rate Selection**

The DTS **may** be operated at fixed speed with autobaud or non-autobaud waveforms.

If transmission speed is varied an autobaud waveform such as STANAG 4539 and STANAG 5069 **shall** be used. If a non-autobaud waveform such as STANAG 4285 is used, the speed **shall not** be varied.

Because the receiver can adapt to transmission speed and interleaver selected, the model is that for each transmission the sender chooses the parameters to use. It can use a number of factors to make this choice including:

1. Receiver Recommendation. This is an important option, as the receiver can generally measure conditions and make a more effective analysis and choice than the sender can using SNR and other measurements. Mechanisms for communicating this recommendation are specified in Section C.7.6.3.

2. Local conditions, including SNR from recent receptions and SNR variation. This can be helpful, noting that sender conditions can be significantly different to receiver.

3. Frame Error Rate from recent transmissions to the peer.

4. Communication results of communication with other peers.

5. QoS requirements of applications. This may be inferred (e.g., that large volumes of data to be transferred requires optimization for throughput and small volumes (likely to be application acks and XMPP chat type applications) should be optimized for latency. This may be known, based on QoS parameters associated with each SAP. The primary choice is to optimize for throughput or latency.

Optimizing for throughput will typically mean choosing an aggressive speed with long interleavers and accepting a high frame error rate. Optimizing for latency will typically mean choosing a conservative speed and perhaps repeating D_PDUs.

6. Choice of peers. A transmission may have data intended for multiple peers to receive. In this situation, recommendations from each peer **shall** be taken into consideration.

The algorithms and choices made are left to the implementation. At a minimum, using receiver recommendations and basic QoS analysis is **recommended**.

C.7.6.2.1. Bandwidth Choice for STANAG 5069

STANAG 5069 wideband HF waveform requires specification of a bandwidth (3kHz – 48kHz). Two approaches are supported by this standard:

1. Fixed Bandwidth. The bandwidth is configured to a fixed value for a given peer or the whole network.

2. Use of 4G ALE as specified in MIL-STD-188-141D. The STANAG 5069 design is intended for use with 4G ALE and this is the approach that is anticipated will generally used. Use of STANAG 5069 generally needs to adapt bandwidth to conditions.

C.7.6.3. Receiver Recommendations

The receiver of a transmission is much better placed than the transmitter to determine best transmission speed and related settings. This is because is has access to SNR and other signal-related information on the transmissions received from a sender.

This specification defines a number of EOW messages that the receiver uses to communicate its recommendations to the sender. Because EOWs are part of the standard D_PDU header, EOWs can be sent by a receiver without incurring any protocol overhead.

When a node transmits receiver recommendations, it may send recommendations to multiple nodes in one transmission. These EOW **shall** only be used in D_PDUs sent to a unicast address and the recommendation applies to transmissions from the node identified by that unicast address.

C.7.6.3.1. Standard Operation

The EOWs described in this section **shall not** be transmitted to Edition 3 peers.

Three of the five EOWs are simple communication of receiver recommendations to the sender.

1. MAX-SPEED (TYPE 8) EOW is specified in Section C.6.5. It specifies the recommended transmission speed to achieve maximum throughput and the minimum length of interleaver to be used.

2. LOW-SPEED (TYPE 9) EOW is specified in Section C.6.6. It specifies the maximum recommended speed for low latency traffic and the minimum length of interleaver to be used

3. MAX-SEGMENT (TYPE 10) EOW is specified in Section C.6.7. It specifies a maximum recommended C_PDU Segment Size. This is particularly useful for fixed speed networks, where reducing the size of D_PDUs transmitted is the only option to adapt to poorer conditions.

These EOWs **may** be used by a receiving node to communicate information to the sender to facilitate better speed selection.

SPEED-USED (TYPE 12) EOW is specified in Section C.6.9. SPEED-USED EOW is used to communication transmission speed and interleaver from a sender to a receiver. It is useful when a receiver cannot determine this information locally. The information in this EOW **shall** apply to the current transmission only. It is recommended to only send this EOW when it is determined that the receiver needs it.

SENDER-APPROACH (TYPE 11) EOW is specified in Section C.6.8. SENDER-APPROACH EOW is used to communicate node capabilities to a peer to which it is sending data. There are three pieces of information conveyed:

1. Information as to whether transmission is fixed or variable speed.
2. The strategy the sender is currently using.
3. Whether the node can determine modem transmission speed, to communicate the need for the peer to send SENDER-SPEED EOW.

The first two elements of SENDER-APPROACH EOW facilitate the receiver to prioritize which EOWs to send. Prioritization may mean only sending one type, or repeating transmission of some EOW types more. For example if bulk data is being transferred, the MAX-SPEED EOW is likely to be the most important EOW. Use of SENDER-APPROACH EOW will be most important at low speeds or for short transmissions where it may only be possible to send a limited number of EOWs. At fixed speed, only the recommended C_PDU Segment Size is useful. When choosing the settings, the sender should consider which EOWs it is most interested to receive.

Table C-25 sets out the recommended receiver interpretation of the SENDER-APPROACH EOW value. Priority indicates choice of use when slots are limited and for repeats.

Approach Setting	MAX-SPEED EOW	LOW SPEED EOW	MAX-SEGMENT EOW
Fixed Speed	Do Not Use	Do Not Use	Use
Variable + Bulk (00)	Use (top priority)	Do Not Use	Use
Variable + Low Latency (01)	Do Not Use	Use (top priority)	Use
Variable + Mixed (10)	Use (top priority)	Use (top priority)	Use
Variable + Intermediate (11)	Use (top priority)	Use	Use

Table C-25. Rules for Interpreting SENDER-APPROACH EOW

C.7.6.3.2. **Operation With Edition 3 Peers**

To communicate receiver recommendations to a peer that supports Edition 3, the ED3-BASIC-RATE (Type 1) EOW defined in Section C.6.2 **shall** be used. The ED3-BASIC-RATE EOW **shall not** be sent to peers known to support Edition 4 (or subsequent).

There are two key recommendations that can be derived from this EOW:

1. Transmission speed, with options for standard HF data rates. This speed should be treated as a recommended speed for maximum throughput and as a recommendation to not use a faster speed.

2. Interleaver. This should be treated as a recommended interleaver to use

C.7.7. **Service Mapping onto Modem & TRANSEC Services**

The Data Transfer Sublayer uses services from the layers below. The layer immediately below is the MAC layer. The MAC layer provides functions to control access to the channel. The interface to Modem and TRANSEC is specified as part of DTS, as the DTS is the primary user of these services. The MAC layer relationship to the DTS varies with the mechanism:

1. For CSMA (Annex K), the MAC layer simply manages timers that control access.

2. For WTRP (Annex L), the MAC layer has protocol, but this protocol uses the DTS peer protocols for data transfer of M_PDUs.

This layer below DTS/MAC may be one of two types of entity:

1. A Modem for HF or other frequencies; or

2. A TRANSEC device/layer which provides data encryption/decryption of all data transferred. This TRANSEC capability may also be used to provide COMSEC protection of user data,

There will be a direct data interface to the layer below, and three options to provide it are described in the following sections.

C.7.7.1. **Modem Control Interface**

The DTS can operate at fixed speed without access to modem control.

In order to change transmission speed and interleaver as specified in Section C.7.6, the DTS needs to be able to control the modem. In order to be able to make recommendations on transmission parameters as specified in Section C.7.6.3, a node needs access to SNR and other information from the modem.

A control connection from DTS to Modem is needed in order to achieve this. Where TRANSEC is used, this control connection needs to use a Crypto Bypass in order to communicate directly with the modem.

Modem control interfaces are not specified in STANAG 5066.

C.7.7.2. **ALE Unit Control Interface**

ALE Units are often provided in conjunction with HF Modem products. Control of ALE Units comes from two parts of the STANAG 5066 stack:

1. The Channel Access Sublayer controls use of 1:1 ALE. This access can be considered to pass down through the DTS.
2. MAC layer controls use of multi-node ALE.

A control connection from STANAG 5066 Stack to ALE Unit is needed in order to support these functions. Where TRANSEC is used, this control connection needs to use a Crypto Bypass in order to communicate directly with the modem.

ALE Unit control interfaces are not specified in STANAG 5066.

C.7.7.3. Data Interfaces

C.7.7.3.1. Data Direct to Modem

STANAG 5066 DTS can connect directly to a modem.

STANAG 5066 does not specify or require an interface here. MIL-STD-188-110D Appendix A specifies a protocol for data communication with an HF modem over TCP. Use of this protocol in conjunction with STANAG 5066 to interface to a modem is **recommended**.

C.7.7.3.2. Data to TRANSEC or Modem using Annex D

STANAG 5066 Annex D ("Interface between Data Transfer Sublayer and Communications Equipment") defines an synchronous serial interface that can be used to connect directly to a modem or to a crypto device providing TRANSEC.

When using this interface for the DTS, the follow considerations are noted:

1. Because there is no clear start/stop timing for this interface, the DTS will generally need to pad the start and end of transmissions, in order to avoid data loss.

2. Because synchronous serial interfaces are bit-aligned, a receiving DTS **shall** treat an inbound data stream as a bit-stream and make use of the Maury-Styles D_PDU header to determine the byte alignment required by the DTS protocols

C.7.7.3.3. Data to TRANSEC using Annex T

STANAG 5066 Annex T ("TRANSEC Crypto Layer using AES and other Protocols") defines a TRANSEC layer with a peer communication protocol. The DTS can use this directly, without need for padding. The protocol specified is byte-aligned.

STANAG 5066 does not standardize an interface to the TRANSEC layer specified in Annex T.

C.8. Extended D_PDU Summary

Annex L (HIGH-FREQUENCY WIRELESS-TOKEN-RING-PROTOCOL (WTRP) REQUIREMENTS) has assigned Extended D_PDUs with values in range 1-4. These are all used for control purposes. The state machines in this Annex have been written to support this usage.

Extended D_PDU Types 0, and 5-255 are reserved for use in future editions of STANAG 5066. Note that new assignments **may** require changes to the state tables

of this annex.

C.9. Deprecated Services

Two services specified in Edition 3 are optional in this edition and deprecated. It is anticipated that these services will be removed in future updates of this specification. They are retained to ensure interoperability with Edition 3 systems, although it is believed that such use is minimal.

Specification of these services is primarily by reference to the text in Edition 3, which ensures full alignment.

C.9.1. High Data Rate Change Request PDU

This optional deprecated service is defined as the HDRCR profile option.

This PDU and its use are specified in Section C.5.5 of Edition 3.

C.9.2. Expedited Data (User Service)

The DTS sublayer defines transfer of Expedited Data, which is used by the CAS sublayer. In Edition 3, the SIS layer specifies service elements that handle Expedited data. This service is retained in Edition 4, but is optional and deprecated.

This optional deprecated service is defined as the EXPEDITED DATA profile option.

Edition 4 DTS specifies only handling of Normal User Data. Handling of Expedited User Data is very similar, but it uses the DTS expedited service. The Edition 3 DTS specification makes clear the details of this mapping, and is the reference for this usage.

C.10. Changes in Edition 4

This section describes changes in edition 4 Annex C, relative to edition 3 of STANAG 5066.

1. Operation over duplex links is made clear. A model is introduced, distinguishing single channel, duplex and broadcast operation. Sections are added to clarify operation.

2. Duplex used only for duplex operation. Edition 3 used the term Duplex to describe data flowing in both directions, which could be over duplex or half duplex link. This confusing usage removed.

3. Data rate adaption for Ed4 peers is only supported for auto-baud waveforms such as STANAG 4539 or STANAG 5069. Fixed speed operation over non-auto-baud waveforms is supported. Rationale for dropping the Edition 3 Data Rate Change mechanism that supports non-autobaud waveforms:

    - The Edition 3 mechanisms for data rate change are complex and fragile.
    - Data rate change with non-autobaud waveforms is not needed in modern deployments.
    - It is problematic to specify duplex operation of these mechanisms or use of these mechanism with ALE, both of which are key elements of Edition 4.
    - Many Edition 3 implementations do not support this option.

4. Optional interworking with Ed3 peers using Ed3 stop/start mechanism and Ed3 Annex I frequency change is specified.

5. A number of new capabilities are introduced. These are added so that they will only be used in communication with peers that are known to support edition 4 or a subsequent edition. Mechanisms are defined to determine this support. The new mechanisms **shall not** be used with peers that are known to not support edition 4 or where peer capability is unknown. This ensures robust interoperability with older systems.

6. A padding D_PDU is introduced, which can increase resilience and performance by adding EOTs and EOWs to a transmission in space that could not otherwise be utilized.

7. A data rate selection process has been added, more appropriate to auto-baud waveforms. Interoperation with edition 3 is specified, as well as edition 4-specific capabilities that include support for WBHF speed recommendations/

8. Increased flexibility on D_PDU size choice which will improve performance, resilience and rapid handling of higher priority data.

9. Use of 16 bit frame sequence number, which is important to achieve good ARQ performance at WBHF and faster narrowband speeds.

10. An extension D_PDU that allows additional D_PDU types to be defined. This is used in Edition 4 Annex L and enables other new capabilities to be added.

11. Explicit prevention of in-order delivery for non-ARQ, as this can lead to system lock-up when data is lost.

12. Incorporation of implementation notes for STANAG 5066 Edition 3 Annex H (Implementation Guidance and Notes) at relevant points, with updates and extended notes based on more recent experience.

13. Folding in technical elements (EOW additional values) and implementation notes from STANAG 5066 Edition 3 Annex G (Use of Waveforms at Data Rates Above 2400 bps).

14. New section clarifying D_PDU transmission rules.

15. New section clarifying mappings onto Modem and TRANSEC.

16. Clarification of use of ALE with reference to Edition 4 changes in Annex B and Annex J.

17. The High Data Rate Change Request PDU and associated EOW are deprecated. The functionality is replaced with a pure EOW approach for Ed4 peers. It is not believed to be needed for Ed3 interoperability, but retained in case it is.

18. The Non-ARQ with errors service is made optional (mandatory in Ed3).