NATO STANDARD

AComP-5066

TECHNICAL STANDARDS FOR HF RADIO LINK LAYER AND APPLICATION SUPPORT PROTOCOLS FOR SINGLE CHANNEL WAVEFORMS

Edition A, Version 1

November 2021

NORTH ATLANTIC TREATY ORGANIZATION

ALLIED COMMUNICATIONS PUBLICATION

Published by the
NATO STANDARDIZATION OFFICE (NSO)

INTENTIONALLY BLANK

NORTH ATLANTIC TREATY ORGANIZATION (NATO)

NATO STANDARDIZATION OFFICE (NSO)

NATO LETTER OF PROMULGATION

[Day] [Month] 2021

1. The enclosed Allied Communications Publication AComP-5066, Edition A, TECHNICAL STANDARDS FOR HF RADIO LINK LAYER AND APPLICATION SUPPORT PROTOCOLS FOR SINGLE CHANNEL WAVEFORMS, which has been approved by the nations in the C3 Board, is promulgated herewith. The agreement of nations to use this publication is recorded in STANAG 5066.

2. AComP-5066, Edition A, Version 1 is effective upon receipt.

3. This NATO standardization document is issued by NATO. In case of reproduction, NATO is to be acknowledged. NATO does not charge any fee for its standardization documents at any stage, which are not intended to be sold. They can be retrieved from the NATO Standardization Document Database ((https://nso.nato.int/nso/) or through your national standardization authorities.

4. This publication shall be handled in accordance with C-M(2002)60.

Dimitrios SIGOULAKIS
Major General, GRC (A)
Director, NATO Standardization Office

INTENTIONALLY BLANK

RESERVED FOR NATIONAL LETTER OF PROMULGATION

INTENTIONALLY BLANK

RECORD OF RESERVATIONS

CHAPTER	RECORD OF RESERVATION BY NATIONS
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Document Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

RECORD OF SPECIFIC RESERVATIONS

[nation]	[detail of reservation]
Note: The reservations listed on this page include only those that were recorded at time of promulgation and may not be complete. Refer to the NATO Standardization Document Database for the complete list of existing reservations.	

INTENTIONALLY BLANK

TABLE OF CONTENTS

INTENTIONALLY BLANK

TECHNICAL STANDARDS FOR HF RADIO LINK LAYER AND APPLICATION SUPPORT PROTOCOLS FOR SINGLE CHANNEL WAVEFORMS

AIM

1. The aim of this agreement is to define the functions and interfaces required for networked, error- free communication over HF radio channels, nominally for beyond-line-of-sight communications. This STANAG makes use of waveforms standardized in other STANAGs and standardized applications. This STANAG defines the necessary intermediate layers to provide a complete solution.

AGREEMENT

2. The participating nations agree to implement the protocols and interfaces defined in this STANAG (including mandatory Annexes) to provide communications over HF radio channels.

GENERAL

3. STANAG 5066 describes a set of functions, segregated logically into layers, together with the interfaces, data formats, and procedures required for interoperability over HF radio. It does not contain performance specifications.

This document is organised so that the main body gives an overview of the structure of the STANAG and the capabilities that should be realised when it is implemented. Annexes A, B, C, D, F and S describes the mandatory interfaces, data formats and procedures. Annexes O, P, Q, U and V contains applications supported. Applications Support. Annexes J, K, L, N, R and T are required for specific profile options. Annexes E, G, H, I and M are intentionally blank whose content from Edition 3 have either been moved to other annexes or deleted.

IMPLEMENTATION OF THE AGREEMENT

4. This STANAG is implemented by a nation when data communication on long-haul or short-haul HF radio channels complies with the characteristics detailed in this agreement.

DEFINITIONS

Node — An implementation of the profile described in the main body of and mandatory annexes to this STANAG. The node is generally assumed to

include the HF (modem and radio) and cryptographic equipment required for communications.

Subnetwork A collection of nodes. As a whole, a subnetwork provides a reliable networked data-transport service for external users or clients.

1. INTRODUCTION

This standard specifies protocols for data communication over HF radio, which will usually be used for beyond- line-of-sight communication. This standard describes a set of functions, segregated logically into layers, together with the interfaces, data formats, and procedures required for interoperability. External standards and specifications are referenced and used where appropriate. The technical characteristics that are required to ensure interoperability and reliable system operation are described in the main body of and mandatory annexes to the document. Information-only annexes provide information on possible implementation of interfaces and subnetwork clients. The annexes also contain implementation advice based on experience during the development and deployment of the protocols.

This document is organised so that the main body gives an overview of the structure of the standard and the capabilities that should be realised when it is implemented. The details of the interfaces, data formats, and procedures are described in a number of mandatory and informational annexes.

1.1. Changes in This Edition

The document structure has been fundamentally updated to match current NATO standardization guidance as codified in AAP-03. The STANAG document is now a cover document as Edition 4. This technical Allied Communications Publication (AComP) is a new document as Edition A, which is associated with the STANAG document. If and when changes to mandatory functionality are made in the standard, the AComP document will be updated to Editions B, C, and so on, while the parent STANAG will be simultaneously updated to Editions 5, 6, and so on.

The primary driver for functional changes in this edition is to provide support for Wideband HF and ALE. A number of small improvements have been made as well as general clean up of the specification text.

Functional changes to STANAG 5066 are set out in Section 9. The introductory text has been substantially rewritten to improve clarity.

2. SERVICES PROVIDED

This STANAG enables interoperability at the two major classes of interface: first, the “common air interface”, describing how information is exchanged between nodes by radio; and second, the non-HF interfaces which allow external users or clients to interact with the subnetwork and with each other over the subnetwork. While physical interfaces are left up to the system implementer (e.g., Ethernet, FDDI, internal bus, or shared memory), the data formats (primitives) and procedures that make up the interface are specified in detail so that client applications can make use of the subnet.

2.1. Common Air Interface: Reliable Data Communications over HF Radio

Reliable data communications over HF radio is provided by using an ARQ data link protocol supported by the modern waveforms referenced in this STANAG. The STANAG is suitable for use with other waveforms with similar characteristics.

The data transfer sublayer defined in the profile supports automatic changes of the user data rate (that is, code rate) of the HF modem in response to changing channel conditions (adaptive data rate). This capability requires remote control of the HF modem and use of auto-baud waveforms such as STANAG 4539 or STANAG 5069. The profile is defined so nodes in which remote control of the modem, and hence adaptive data rate, is not available will interoperate with nodes which do have the capability.

The profile defined here supports frequency selection by use of Automatic Link Establishment (ALE), but does not require that ALE is available.

2.2. The STANAG 5066 Protocol Stack

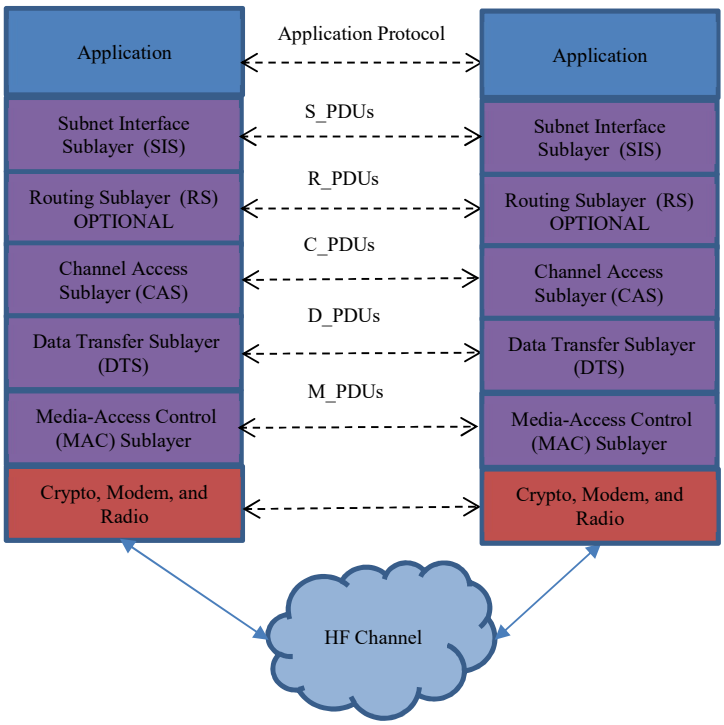


Figure 1: STANAG 5066 Layer Architecture

STANAG 5066 provides and interface between applications and the HF channel communications, comprising Crypto, Modem and Radio. The interface to the HF

Channel is described in Section 3. Applications are described in Section 4.

STANAG 5066 has a layered architecture, with peer protocol defined with a set of PDUs and service interfaces specified between each layer. The following sublayers are specified:

1. The **Subnetwork Interface Sublayer (SIS)** provides a service interface to applications using STANAG 5066. Annex A defines the service interface provided by SIS to applications. The primary service function is to transfer blocks of data (UNIDATA). The SIS defines a peer protocol using S_PDUs.

2. The **Routing Sublayer (RS)** is an optional sublayer that is used in configurations where data needs to traverse an HF channel multiple times. This is needed to support Wireless Token Ring Protocol (WTRP) configurations with partial connectivity. The RS is specified in Annex R.

3. The **Channel Access Sublayer (CAS)** controls communication between peers by managing “physical links” which control ARQ communication between peers. The CAS can provide simultaneous access to multiple peers or constrain operation to one active peer at a time. The CAS is specified in Annex B.

4. The **Data Transfer Sublayer (DTS)** provides data transfer with one or more peers. The DTS provides a reliable (ARQ) data link service, as well as best-effort (non-ARQ) service for broadcast, multicast and communication to peers in EMCON. The DTS supports data rate selection for STANAG 4539 and STANAG 5069 waveforms. The DTS is specified in Annex C.

5. The **Media-Access-Control (MAC) Sublayer** provides mechanisms for enhanced media-access control capability for HF data communication in multi-node networks. Whereas the Channel-Access Sublayer provides pairwise logical link control mechanisms to establish a point-to-point link (or set of multiple, independent point-to-point links) for data communication, the Media-Access- Control Sublayer introduces modes for enhanced media-access control capability for HF data communication in multi-node networks, and the prescribed method in which they may be used with other STANAG 5066 capabilities. These optional channel-access modes extend or modify, but do not replace, the pairwise logical link control mechanisms defined for the Channel Access Sublayer. General requirements for the Media-Access-Control Sublayer are defined in Annex J of this STANAG, with requirements on each of the defined protocols for media-access- control provided in Annexe K for Carrier-Sense-Multiple Access and Annex L for Wireless-Token- Ring Protocol (WTRP). Annexes may be added in future versions of this STANAG for other techniques, such as Adaptive Time Division Multiple Access.

3. STANAG 5066 Interfaces to the HF Channel

STANAG 5066 defines three ways to interface to the HF channel, which are described in this section. These interfaces can all be used with the waveforms defined in STANAG 4285, STANAG 4529, STANAG 4539 and STANAG 5069. It is anticipated that STANAG 5066 can be used with other waveforms with similar characteristics to these.

3.1. Direct Interface to Modem

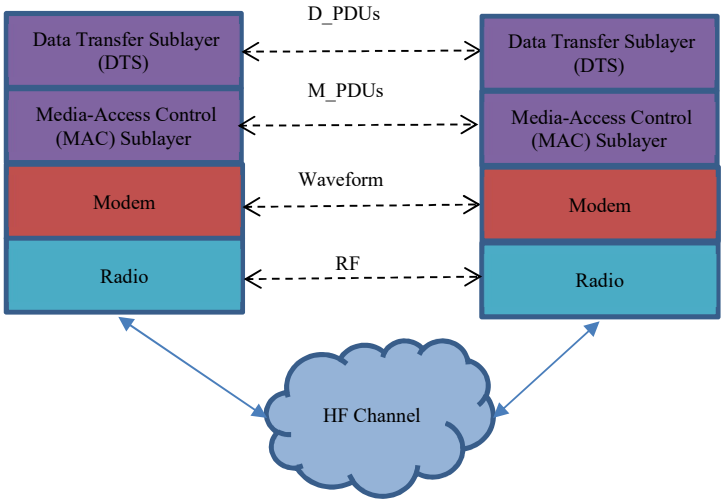


Figure 2: STANAG 5066 Direct Interface to Modem

The STANAG 5066 stack **may** interface directly to a modem, with a stack as shown in Figure 2. Data interface between STANAG 5066 stack and Modem **may** use the synchronous serial interface specified in Annex D. This data interface **may** use other communication mechanisms such as the TCP data interface specified in Annex A of MIL-STD-188-110D.

3.2. Interface to Synchronous Serial Crypto Equipment

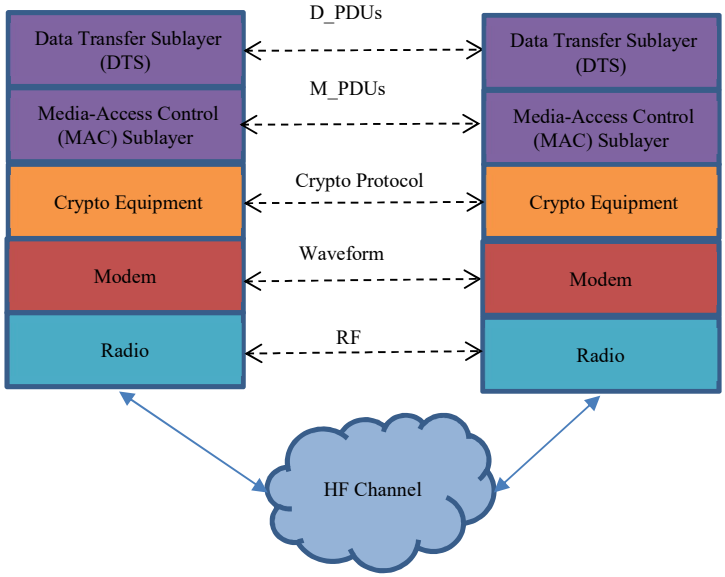


Figure 3: STANAG 5066 Interface to Synchronous Serial Crypto Equipment

The STANAG 5066 stack can be interfaced to Synchronous Serial Crypto Equipment, such as BID-950, KG-84C and KIV-7, as shown in Figure 3. Data interface between STANAG 5066 and Crypto Equipment **shall** conform to Annex D of this specification.

3.3. Interface using AES and other Encryption Algorithms

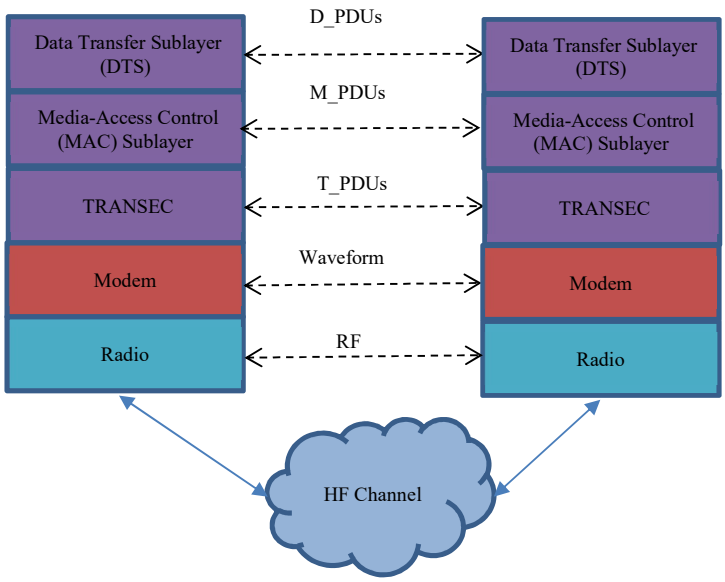


Figure 4: STANAG 5066 Interface using AES and other Encryption Algorithms

The STANAG 5066 stack can be extended downwards to include encryption in a TRANSEC sublayer following the specification of Annex T of this specification. Annex T specifies use of the Advanced Encryption Standard (AES) and provides a general framework for use of alternative encryption algorithms.

Data interface between the TRANSEC layer and Modem **may** use the synchronous serial interface specified in Annex D. This data interface **may** use other communication mechanisms such as the TCP data interface specified in Annex A of MIL-STD-188-110D.

3.4. Modem Control and Monitoring

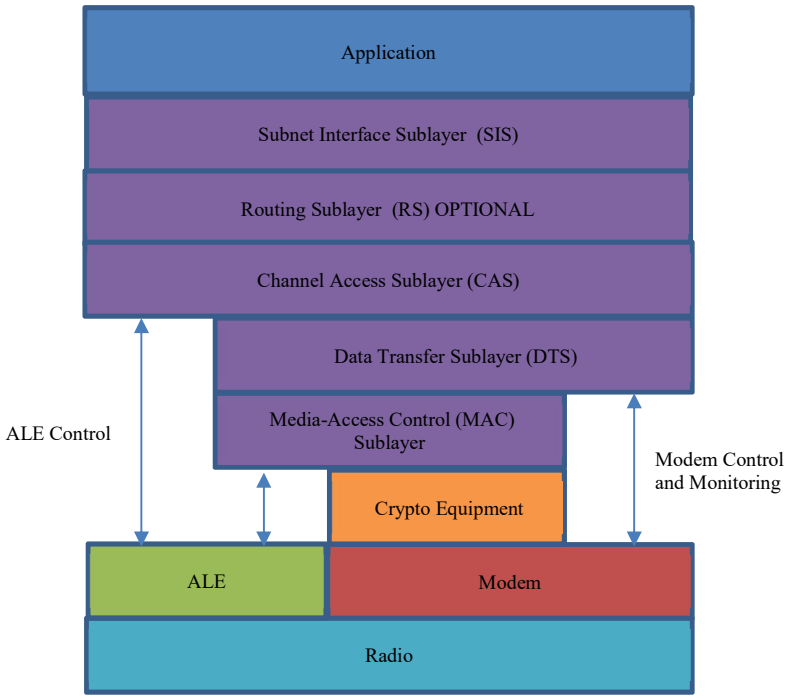


Figure 5: STANAG 5066 Modem and ALE Control

STANAG 5066 allows for, but does not require control communication with the modem. Monitoring and control of the modem is performed by the DTS, as shown in Figure 5. The following requirements are identified:

1. To achieve Data Rate Selection, as described in Annex C, the sender needs to control the modems sending speed and associated parameters such as interleaver.

2. To provide optimum recommendations on transmission parameters, the receiver needs to be able to monitor SNR and other receive characteristics.

3. To determine switching time when no EOT is received, performance of Annex K and Annex L can be optimized by monitoring end of transmission at modem level.

In addition to these requirements and implementation may use additional modem control, configuration and monitoring capabilities to improve system performance and management.

3.1. Use of Automatic Link Establishment (ALE)

Automatic Link Establishment (ALE) enables selection of best frequency. For STANAG

5069, 4G ALE also allows selection of transmit and receive bandwidth. ALE mechanisms are specified in MIL-STD-188-141D, providing 2G (synchronous), 3G (asynchronous) and 4G mechanisms. STANAG 4538 defines 3G mechanisms.

ALE sits at the same level as the modem in the protocol architecture. ALE and Modem share use of the radio, with only one of them having access at any moment. ALE and Modem will co-ordinate sharing of the modem.

STANAG 5066 specifies two uses of ALE, as shown in Figure 5:

1. ALE used by MAC Sublayer, as specified in Annex J. In this mode, the MAC layer negotiates a frequency for use by all nodes on the subnetwork.

2. ALE used by the CAS, as specified in Annex B. In this mode, ALE is used to connect to one peer node or to a subset of nodes on the network. This will lead to the SIS queueing traffic for other nodes, as specified in Annex A.

ALE **may** be used in a third implicit mode on a two-node network. In this configuration, ALE can be used to set up a link without interaction with STANAG 5066, as there is only one destination.

3.2. Crypto Bypass

It can be seen in Figure 5 that all user data is passed through the Crypto. In order to achieve use of ALE and variable data rate, STANAG 5066 needs to communicate with components on the other side of the Crypto. This communication is called Crypto Bypass.

When Crypto Bypass is used, additional controls and checks **may** be needed on the information flow. Mechanisms to achieve this are outside of the scope of this STANAG.

4. APPLICATION SUPPORT

The SIS service provides a simple service interface to support applications, as defined in Annex A.

4.1. Applications Supported

Each application supported by STANAG 5066 will use a SAP (Subnet Access Point). STANAG 5066 can support up to 16 applications sharing a channel, each with a different SAP ID (0-15).

Annex F specifies SAP ID assignment for a number of applications. Where one of these applications is used, the assigned SAP ID **shall** be used. If an application listed in Annex F is not being used, a deployment **may** use its assigned SAP ID for a different application.

STANAG 5066 fully specifies one application, specifies supporting protocol for other

applications, and references external applications. This includes:

1. HF Operator Chat. A simple operator chat protocol specified in Annex O.

2. Support for ACP 127 legacy formal military messaging using the Character Oriented Stream Service (COSS), specified in Annex P.

3. Support for Multicast Transfer using ACP 142, specified in Annex Q. This enables two services over ACP 142:

    a. STANAG 4406 Annex E, to provide formal military messaging.

    b. Multicast Email (MULE) specified in RFC 8494, to provide email or formal military messaging using RFC 6477.

4. Compressed File Transfer Protocol (CFTP) which can be used to provide basic email services, specified in Annex V.

5. Chat and Presence using XMPP, specified in XEP-0365.

6. IP Client, that enables an IP subnet to be operated over STANAG 5066. This provides support for some IP services, such as ICMP Ping. IP Client is specified in Annex U.

4.2. Enabling Applications to Share a STANAG 5066 Stack

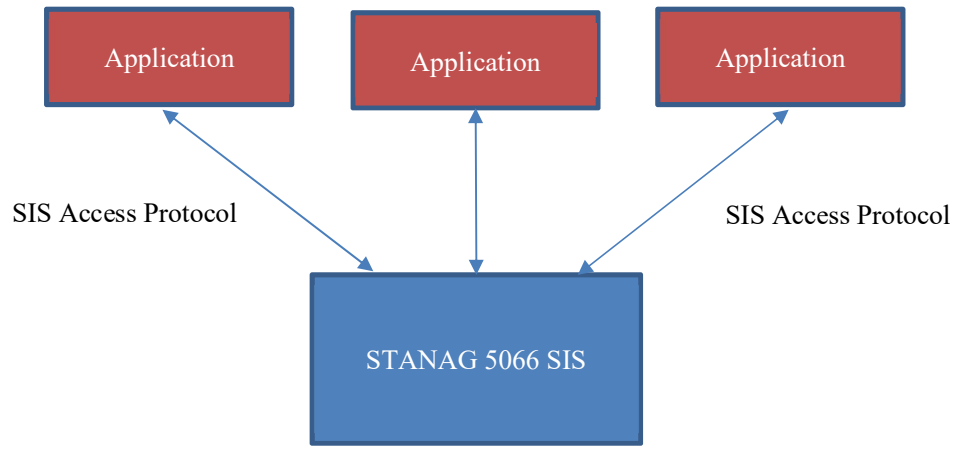


Figure 6: SIS Access Protocol Model

It is desirable to have multiple applications share a single STANAG 5066 Stack, so that a common stack implementation can be used and that data from the different applications can be multiplexed over STANAG 5066 communication. This is achieved

by use of the SIS Access Protocol, illustrated in Figure 6 and specified in Annex S. The SIS Access Protocol operates locally to a node and has no end to end implications.

5. List of Annexes

Annex A: Subnetwork Interface Sublayer (mandatory)
Annex B: Channel Access Sublayer (mandatory)
Annex C: Data Transfer Sublayer (mandatory)
Annex D: Interface between Data Transfer Sublayer and Communications Equipment (mandatory)
Annex E: Absent
Annex F: SAP Assignment (mandatory)
Annex G: Absent
Annex H: Absent
Annex I: Absent
Annex J: General Requirements for Enhanced Media-Access-Control (MAC) Capabilities in Multi-Node STANAG 5066 Networks
Annex K: High-Frequency Carrier-Sense Multiple-Access (CSMA) Protocols
Annex L: High-Frequency Wireless Token-Ring-Protocol (WTRP) Requirements
Annex M: Reserved
Annex N: Guidance on Address Management in STANAG 5066 Networks
Annex O: HF Operator Chat
Annex P: ACP 127 & Character-Oriented Serial Stream
Annex Q: ACP 142
Annex R: Routing Sublayer
Annex S: SIS Access Protocol (mandatory)
Annex T: STANAG 5066 TRANSEC Crypto Sublayer using AES and other Protocols
Annex U: IP Client
Annex V: Compressed File Transfer Protocol

Annexes not marked as “mandatory” are “information only”.

Annexes marked “Absent” were present in previous editions of STANAG 5066. Annex M is marked “reserved” as a provisional allocation has been made.

6. Interoperability with Edition 3

Edition 4 of STANAG 5066 introduces a number of new protocol elements to improve the CAS and DTS services. 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. Use of the Capability EOW defined in Section C.6.4 of Annex C. 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.

7. Profiles

STANAG 5066 is a complex standard with many options. This section sets out profiles of the layer services and applications provided using STANAG 5066. This enables a deployment or a procurement to clearly specify what is needed from STANAG 5066.

In addition to the layer services and applications, a broader profile might include Waveforms and ALE protocols used along with other interoperability choices of layers below STANAG 5066. Such a profile is outside the scope of this STANAG.

7.1. Layer Services Profile Options

STANAG 5066 contains a number of options in the core protocols and functions set out in Annexes A, B, C and J contain some optional elements, listed in Table 1 as profile options. This table provides an identifier for each option which are referenced in the Layer Services Profile. The table references where in the STANAG each of these options are specified.

Profile Option	Annexes	Notes
HARD LINKS	A, B	DEPRECATED. Retained for Edition 3 compatibility. Not expected to be used.
EXPEDITED DATA	A, C	DEPRECATED. Retained for Edition 3 compatibility. Not expected to be used.
NON-ARQ WITH ERRORS	A, C	Service defined in Annex A, with implementation in Annex C. Not used by any current applications, but may be used in the future.
DUPLEX FIXED	B, C	Duplex operation with a fixed frequency two node network
MULTIPLE ACCESS	B	Operation with Multiple Simultaneous Peer Access model, necessary to support multicast with non-ALE networks with three or more nodes using CSMA or WTRP
ALE 1:1	B	Use Single Peer Access Model in conjunction with ALE. This is the ALE mode that is expected to be most commonly used. This option has two modes “EXPLICIT CAS-1” and “IMPLICIT CAS-1”. One of these modes needs to be selected.
MULTICAST ALE	B	Use of Multicast ALE to enable support of multicast with ALE. This might help improve performance for multicast applications over network not using CSMA or WTRP
DUPLEX WITH ALE	B, C	Use of ALE with Duplex, to enable Single Peer Access Model in a multi-node network. This would be used to support a network where multiple nodes have duplex capability.
HDRCR	C	DEPRECATED. Retained for Edition 3 compatibility. Use of the High Data Rate Change Request PDU. Not expected to be used.
MAC ALE	J	Use of ALE with MAC ALE

Table 1: Layer Services Profile Options

7.2. Layer Services Profile

This section specifies a set of Layer Service Profile for STANAG 5066. The following Annexes are mandatory in all of the profile options.

Annex A: Subnetwork Interface Sublayer
Annex B: Channel Access Sublayer
Annex C: Data Transfer Sublayer
Annex D: Interface between Data Transfer Sublayer and Communications Equipment
Annex S: SIS Access Protocol

The following Annexes are required in some of the profile options:

Annex J: General Requirements for Enhanced Media-Access-Control (MAC) Capabilities in Multi-Node STANAG 5066 Networks
Annex K: High-Frequency Carrier-Sense Multiple-Access (CSMA) Protocols
Annex L: High-Frequency Wireless Token-Ring-Protocol (WTRP) Requirements

The following Annexes are optional, and if their use is required it **shall** be specified in conjunction with profile:

Annex R: Routing Sublayer
Annex T: STANAG 5066 TRANSEC Crypto Sublayer using AES and other Protocols

Both of these annexes can be used in conjunction with any profile. Annex T is an alternative to Annex D, and this **may** become a choice in future versions of this standard. Annex R is important to support some WTRP configurations, and so is explicitly noted with these profile options. Annex R **may** be used with any profile option.

Table 2 defines a list of profiles. For a given deployment, every node needs to follow the same profile, as this STANAG does not provide a dynamic mechanism to distinguish between these profiles. When procuring systems, it can be sensible to specify multiple profiles, to enable deployment with different profiles.

Note that the Non-ARQ with Errors profile option is not used in any profile, as there are no standardized or open protocols using this option. If future applications are defined that need this service, use of the “Non-ARQ Applications” profile option will be added to profile options in a future version of this standard.

Profile	Mandatory Profile Options and Annexes	Notes
ALE Basic (IMPLICIT CAS-1)	ALE 1:1 (mode IMPLICIT CAS-1)	Profile suitable for Skywave operation where ALE is always used. This mode optimizes for short lived ALE connections/soft links.
ALE Basic (EXPLICIT CAS-1)	ALE 1:1 (mode EXPLICIT CAS-1)	Profile suitable for Skywave operation where ALE is always used. This mode optimizes for long lived soft links where multiple ALE links are used support one soft link.
Duplex Pair	DUPLEX FIXED	A profile for use between a pair of nodes using duplex and fixed frequency.
Duplex Network	DUPLEX WITH ALE; ALE 1:1	A profiled for use to support a network with multiple nodes where some or all nodes are capable of duplex operation ALE 1:1 Mode needs to be specified in conjunction with this mode.
Multicast ALE Network	MULTICAST ALE; ALE 1:1	A profile to support an ALE network which can optimize for multicast applications, which has potential to improve performance. ALE 1:1 Mode needs to be specified in conjunction with this mode, to support ARQ and Unicast applications.
Non-ARQ Applications	NON-ARQ WITH ERRORS; MULTIPLE ACCESS;	A profile suitable to support (non-standard or future) applications using the Non-ARQ with errors service
CSMA Fixed Frequency	MULTIPLE ACCESS, Annex J, Annex K	Multi-node using Annex K Useful for Surface Wave with light load or Skywave deployment without ALE
CSMA with ALE	MULTIPLE ACCESS, Annex J, Annex K, MAC ALE	Multi-node using Annex K with ALE Useful to support Surface Wave deployment with light load where fixed frequency is not appropriate
WTRP Fixed Frequency	MULTIPLE ACCESS, Annex J, Annex L	Multi-node using Annex L Useful for Surface Wave with high load
WTRP with ALE	MULTIPLE ACCESS, Annex J, Annex L, MAC ALE	Multi-node using Annex L with ALE Useful to support Surface Wave deployment with high load where fixed frequency is not appropriate

Table 2: Layer Services Profiles

7.3. Application Profile

When deploying STANAG 5066 systems, it is important that all nodes offer the same set of applications to ensure interoperability. All of the applications standardized by STANAG 5066 or referenced in Annex F, can be used with any STANAG 5066 Layer Services Profile. For this reason, the STANAG Application Profile is specified separately. A full STANAG 5066 profiles comprises both Layer Services Profile and Application Profile.

The Application Profile is simply a list of applications to be used. This list **shall** contain at least one application, which **may** be applications listed in this section or **may** be others applications or a mix. STANAG 5066 has five annexes which specify use of applications operating over STANAG 5066. Most of these annexes reference external standards and only define operation over STANAG 5066:

Annex O:	HF Operator Chat
Annex P:	ACP 127 & Character-Oriented Serial Stream
Annex Q:	ACP 142
Annex U:	IP Client
Annex V:	Compressed File Transfer Protocol

For two of these annexes, the following additional information **shall** be supplied:

1. Annex Q. Whether STANAG 4406 Annex E, RFC 8494 (MULE) or both are supported
2. Annex U. Whether IPv4, IPv6 or both are supported/

The list of applications supported **may** include any application that runs over STANAG 5066. This includes the following applications which are referenced in Annex F:

- HMTP (STANAG 5066 Edition 3 Annex F.5)
- HFPOP (STANAG 5066 Edition 3 Annex F.6)
- XMPP (XEP-0365 “Server to Server communication over STANAG 5066 ARQ”)

8. References

1. NATO STANDARDIZATION AGREEMENT 4203 — Technical Standards for Single Channel HF Radio Equipment

2. NATO STANDARDIZATION AGREEMENT 4285 — Characteristics of 1200/ 2400/ 3600 bps single tone modulators/demodulators for HF radio links

3. NATO STANDARDIZATION AGREEMENT 4406 Annex E — Military Message Handling System, Annexe E: Tactical MMHS Protocol and Profile Solution, March 2005

4. NATO STANDARDIZATION AGREEMENT 4529 — Characteristics of Single-Tone Modulators/Demodulators for Maritime HF Radio Links with 1240 Hz bandwidth

5. NATO STANDARDIZATION AGREEMENT 4538 — Technical Standards for an Automatic Radio Control System (ARCS) for HF Communications Links

6. NATO STANDARDIZATION AGREEMENT 4539 — Technical Standards for Non-Hopping HF Communications Waveforms

7. NATO STANDARDIZATION AGREEMENT 5069 — Technical Standards for Wideband Waveforms for Single Non-Hopping, Flexible-Bandwidth HF Channels

8. DEPARTMENT OF DEFENSE INTERFACE STANDARD 188-110D — Interoperability and Performance Standards for Data Modems, MIL-STD-188-110D, 29 December 2017

9. DEPARTMENT OF DEFENSE INTERFACE STANDARD 188-141D — Interoperability and Performance Standards for Medium and High Frequency Radio Systems, MIL-STD-188-141D, 29 December 2017

10. Combined Communications-Electronics Board ACP127(G), “Tape Relay Procedures”, November 1988

11. Combined Communications-Electronics Board ACP142A, “P_MUL - A Protocol for Reliable Multicast Messaging in Bandwidth Constrained and Delayed Acknowledgement (EMCON) Environments”, October 2008

12. CCITT V41 - Code-independent error-control system, November 1988

13. CCITT V.42 - Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion, March 2002

14. National Institute of Standards and Technology (NIST), “Advanced Encryption Standard (AES)”, 1988

15. REQUEST FOR COMMENTS 791 — Postel, J., “Internet Protocol”, September 1981.

16. REQUEST FOR COMMENTS 6477 — Melnikov, A; Lunt, G., “Registration of Military Message Handling System (MMHS) Header Fields for Use in Internet Mail”, January 2012

17. REQUEST FOR COMMENTS 8200 — Dering, S.; Hinden, R., “Internet Protocol, Version 6 (IPv6) Specification”. July 2017

18. REQUEST FOR COMMENTS 8494 — Wilson, D.; Melnikov, A., “Multicast Email (MULE) over Allied Communications Publication (ACP) 142”, November 2018

19. XMPP Standards Foundation XEP-0365, “Server to Server communication over STANAG 5066 ARQ”, Kille, S., July 2018

9. Changes in Edition 4

Most changes since Edition 3 are in the annexes. Each annex has a section at the end listing changes in Edition 4. The following changes are made in the main document:

1. New Title that better reflects purpose.

2. Updated overall architecture to reflect new annexes.

3. General update of the text to improve clarity.

4. Added section to summarize use of ALE.

5. Added Profiles section.

Table 3 lists the annexes and summarizes the major changes since Edition 3.

Annex	Edition 4 Title	Key Changes
A	Subnetwork Interface Sublayer	Change Hard Links, and Expedited Data to deprecated services, with the expectation of removing these services in future editions., Removal of Rank, which is not used or needed. Moved local service encoding and services without end to end significant to Annex S, which now specifies the SIS protocol.
B	Channel Access Sublayer	Added Edition 4 negotiation and support for ALE.
C	Data Transfer Sublayer	Extended frame sequence number to support higher speeds. Provided D_PDU extensibility.
D	Interface between Data Transfer Sublayer and Communications Equipment	No significant change.
E	Absent	Modem control interface that was out of date and not useful.
F	SAP Assignment	Most content moved to annexes O-Q, S, U and V. Renamed to reflect SAP assignment core content. Some new assignments made.
G	Absent	Requirements on 2400 bps modems, now handled in Annex D.

Annex	Edition 4 Title	Key Changes
H	Absent	Implementation notes. Information still valid folded into Annex D and other places as appropriate.
I	Absent	Frequency change procedure that is no longer needed due to integrated ALE support in Annex C. Ed3 Annex I is referenced in Annex C for backwards compatibility.
J	General Requirements for Enhanced Media-Access-Control (MAC) Capabilities in Multi-Node STANAG 5066 Networks	No significant change.
K	High-Frequency Carrier-Sense Multiple-Access (CSMA) Protocols	Added “slotted” support to optimize performance on small networks.
L	High-Frequency Wireless Token-Ring-Protocol (WTRP) Requirements	Model unchanged. Significant update of protocol.
M	Reserved	
N	Guidance on Address Management in STANAG 5066 Networks	Updated to reflect country address assignment and policy change.
O	HF Operator Chat	Extracted from Annex F. No significant change.
P	ACP 127 & Character-Oriented Serial Stream	Extracted from Annex F. No significant change.
Q	ACP 142	Extracted from Annex F. Description changed (no protocol change) to support MULE as well as STANAG 4406 Annex E.
R	Routing Sublayer	New optional sublayer
S	SIS Access Protocol	Extracted from Annexes A and F. No significant change.
T	STANAG 5066 TRANSEC Crypto Sublayer using AES and other Protocols	New optional layer to provide alternate TRANSEC to Annex D.
U	IP Client	Extracted from Annex F. Added IPv6 support.
V	Compressed File Transfer Protocol	Extracted from Annex F. No significant change.

Table 3: Changes to Annexes

ANNEX A SUBNETWORK INTERFACE SUBLAYER (Mandatory)

This annex defines the interface between the users of the HF subnetwork and the computer information system through which the user accesses the subnetwork.

A.1 Subnetwork Service Definition

A.1.1 Changes in This Edition

The functional differences between this specification and Edition 3 are set out in Section A.5.

The document structure has changed in the following ways:

1. Hard Links. These are now deprecated and specified in Section A.4.1, rather than in Sections A.1 and A.2.
2. Expedited data. This service is now deprecated and specified in Section A.4.2, rather than in Section A.2.
3. Service Encoding. The encoding of service primitives is not used by the peer protocols. It is therefore now specified in Annex S as part of the SIS Access Protocol specification.
4. Local Services. Some services specified in Edition 3 have local significance only and do not impact end to end operation. These services (S_SUBNET_AVAILABILITY; S_DATA_FLOW_ON/OFF; S_MANAGEMENT_MSG; S_KEEPALIVE) are now specified in Annex S rather than in A.2.

A.1.2 Overview

The Subnetwork Service can be used to support multiple applications and multiplexes multiple applications over a single channel. The service provided by the server is application independent and common to all clients irrespective of the task they may perform.

Clients are associated to the Subnetwork Interface Sublayer by Subnetwork Access Points (SAPs). There can be multiple clients simultaneously using the Subnetwork Interface Sublayer. Each SAP is identified by its SAP Identifier (SAP ID). The SAP ID is a number in the range 0-15; hence there can be a maximum of 16 clients using the Subnetwork Interface Sublayer of a single node. SAPs are equivalent to the “ports” of the TCP and UDP protocols.

Clients **may** connect to the Subnet Interface Service using the SIS Access Protocol specified in STANAG 5066 Annex S. Clients **may** use SIS layer directly or connect with a different protocol. Clients are responsible for segmenting larger messages into User Protocol Data Units (U_PDUs).

The Subnetwork Interface Sublayer treats all clients connected to it in the same manner irrespective of the application performed by these clients.

A.1.3 Initiating Data Exchange Sessions

The Subnetwork Interface Sublayer is responsible for initiating the establishment and termination of Sessions with its peers at remote nodes. There are two types of sessions:

- Soft Link Data Exchange Session, which require the making of a point-to-point physical link with a specified remote node.
- Broadcast Data Exchange Session, which provides unreliable transfer of data and does not require the making of a physical link.

Clients for the HF Subnetwork services **may** interleave requests for the various session types in accordance with the capabilities of this standard. Support for only one session type, e.g., restriction to support only a Broadcast Data Exchange Session, **may** be established as part of the local (implementation-dependent) subnetwork management function.

A.1.4 Soft Link Data Exchange Session

The establishment of a Soft Link Data Exchange Session **shall** (1) be initiated unilaterally by the Subnetwork Interface Sublayer which has queued data requiring reliable delivery (i.e., queued ARQ U_PDUs). Soft Links **may** also be initiated for Non-ARQ traffic for use over ALE links, when requested by the Channel Access Sublayer.

The Subnetwork Interface Sublayer **shall** (2) initiate Soft Link Data Exchange Sessions as needed, following the procedure described in Section A.3.2.1.

When all data has been transmitted to a node with which a Soft Link Data Exchange Session has been established, the Subnetwork Interface Sublayer **shall** (3) terminate the Soft Link Data Exchange Session after a configurable and implementation-dependent time-out period in accordance with the protocol specified in Section A.3.2.1.3.

Termination of the Soft Link Data Exchange Session **shall** (4) be in accordance with the procedure specified in Section A.3.2.1.3. The time out period may be zero. The time out period allows for the possibility of newly arriving U_PDUs being serviced by an existing Soft Link Data Exchange Session prior to its termination.

In order to provide “balanced” servicing of the queued U_PDUs, a Soft Link Data Exchange Session **shall** (5) not be maintained for a period which exceeds a specified maximum time if U_PDUs of appropriate priorities are queued for different node(s).

The specified maximum time out period **shall** (6) be a configurable parameter for the protocol implementation. The specific values of the parameters governing the establishment and termination of Soft Link Data Exchange Sessions (e.g. time-out periods etc.) are chosen in the context of a particular configuration (i.e. size of network, etc).

A.1.5 Broadcast Data Exchange Session

The second type of data exchange session is the Broadcast Data Exchange Session. The subnetwork **shall** (1) service only clients with service requirements for non-ARQ U_PDUs during a Broadcast Data Exchange Session. [Note: Clients with service requirements for non-ARQ U_PDUs may be serviced during other session types, however, in accordance with the session’s service characteristics.] A Broadcast Data Exchange Session can be initiated and terminated by a management process, e.g., a local or network administrator management client.

The procedures that initiate and terminate broadcast data exchange sessions **shall** (2) be as specified in Annex C.

A node configured to be a broadcast-only node **shall** (3) use a “permanent” Broadcast Data Exchange Session during which the Subnetwork Interface Sublayer **shall** (4) service no ARQ Data U_PDUs. Alternatively the Subnetwork Interface Sublayer can unilaterally initiate and terminate Broadcast Data Exchange Sessions.

The core broadcast service delivers error free data. There is an optional NON-ARQ WITH ERRORS profile option that allows delivery of data that may contain errors. This service can be useful where it may be preferable to receive data containing errors to receiving no data at all.

A.2 Subnet Service Specification

The Service Interface to the Subnetwork Interface Sublayer provides the interface primitives listed in Table A-1 and defined in the following subsections. The names of these primitives are prefixed with an “S_” to indicate that they are exchanged across the interface between the subnetwork interface sublayer and the application using the service interface. This table is intended to provide a general guide and overview to the primitives. For detailed specification of the primitives, the later sections of this Annex **shall** (1) apply.

Table A-1. Primitives Exchanged with Clients

CLIENT -> SUBNETWORK INTERFACE	SUBNETWORK INTERFACE -> CLIENT
S_BIND_REQUEST (Service Type, SAP ID)	S_BIND_ACCEPTED (SAP ID, MTU)
	S_BIND_REJECTED (Reason)
S_UNBIND_REQUEST ( )	S_UNBIND_INDICATION (Reason)
S_UNIDATA_REQUEST (Destination Node Address, Destination SAP ID, Priority, TimeToLive, Delivery Mode, U_PDU)	S_UNIDATA_REQUEST_CONFIRM (Destination Node Address, Destination SAP ID, Size of confirmed U_PDU, U_PDU)
	S_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, Size of Rejected U_PDU, U_PDU)
	S_UNIDATA_INDICATION (Source Node Address, Source SAP ID, Destination Node Address, Destination SAP ID, Priority, Transmission Mode, *transmission-mode conditional parameters* , U_PDU)

A.2.1 **Management and Flow Control**

It is anticipated that the service interface between the SIS layer and application will contain:

1. Flow Control to control flow of data between SIS and Application.
2. Management information, to provide the application with additional information

These functions are not standardized in this Annex, and **may** be chosen to support the application. These functions are local and do not impact end to end interoperability.

Annex S specifies a management and flow control primitives that are appropriate to use with this annex.

A.2.2 **Content Specification and Use of Primitives**

The content specification and use of the Subnetwork Interface Sublayer primitives **shall** (1) be as specified in the following subsections.

A.2.2.1 **S_BIND_REQUEST Primitive**

Name :
S_BIND_REQUEST ( )
Arguments :
3. SAP ID,
4. Default Service Type

Direction :
Client -> Subnetwork Interface
Description :
The S_BIND_REQUEST primitive **shall** (1) be issued by a new client when it first connects to the subnetwork. Unless this primitive is issued the client can not be

serviced. With this primitive the client uniquely identifies and declares that it is “on-line” and ready to be serviced by the subnetwork.

The first argument of this primitive **shall** (2) be the “*SAP ID”* which the client wishes to be assigned. The SAP ID **shall** (3) be node-level unique, i.e. not assigned to another client connected to the Subnetwork Interface Sublayer for a given node.

The last argument of this primitive **shall** (7) be “*Service Type*” and identifies the default type of service requested by the client and specified in Section A.2.2.6. The *Service Type* argument **shall** (8) apply to all data units submitted by the client unless explicitly overridden by client request when submitting a U_PDU to the subnetwork.

A.2.2.2 S_UNBIND_REQUEST Primitive

Name :
S_UNBIND_REQUEST ( )
Arguments :
NONE
Direction :
Client -> Subnetwork Interface ( )

Description :
The S_UNBIND_REQUEST primitive **shall** (1) be issued by a client in order to declare itself “off-line”. The Subnetwork Interface Sublayer **shall** (2) release the SAP ID allocated to the client from which it receives the S_UNBIND_REQUEST and the SAP_ID allocated to this client **shall** (3) then be available for allocation to another client that may request it.

A client that went off-line by issuing the S_UNBIND_REQUEST primitive can come on-line again by issuing a new S_BIND_REQUEST.

A client can also go off-line by physically disconnecting itself (e.g. powering down the computer which runs the client program) or disconnecting the physical cable (RS232, Ethernet, etc.) which may connect the client to the node.

A.2.2.3 S_BIND_ACCEPTED Primitive

Name :
S_BIND_ACCEPTED ( )
Arguments :
1. SAP ID
2. Maximum Transmission Unit (MTU)

Direction :
Subnetwork Interface -> Client
Description :
The S_BIND_ACCEPTED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer as a positive response to a client’s S_BIND_REQUEST.

The *SAP ID* argument of the S_BIND_ACCEPTED primitive **shall** (2) be the SAP ID assigned to the client and **shall** (3) be equal to the *SAP ID* argument of the S_BIND_REQUEST to which this primitive is a response.

The *MTU* argument **shall** (4) be used by the subnetwork interface sublayer to inform the client of the maximum size U_PDU (in bytes or octets) which will be accepted as an argument of the S_UNIDATA_REQUEST primitive. S_UNIDATA_REQUEST primitives containing U_PDUs larger than the MTU **shall** (5) be rejected by the subnetwork interface. Note that this restriction applies only to U_PDUs received through the subnetwork interface. U_PDUs which are received from the lower HF sublayers (i.e., received by radio) **shall** (6) be delivered to clients regardless of size.

For general-purpose nodes, the MTU value **shall** (7) be 2048 bytes. For broadcast-only nodes, the MTU
shall (8) be configurable by the implementation up to a maximum that **shall** (9) not exceed 4096 bytes.

A.2.2.4 S_BIND_REJECTED Primitive

Name :
S_BIND_REJECTED ( )
Arguments :
1. Reason
Direction :
Subnetwork Interface -> Client
Description :
The S_BIND_REJECTED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer as a negative response to a client's S_BIND_REQUEST. If certain conditions are not met then the Subnetwork Interface Sublayer rejects the client's request.

The *Reason* argument of the S_BIND_REJECTED primitive **shall** (2) specify the reason why the client's request was rejected. Valid *Reason* values **shall** (3) be as specified in the table below.

Reason	Value
Not Enough Resources	1
Invalid SAP ID	2
SAP ID already allocated	3
ARQ Mode unsupportable during Broadcast Session	4

The value assigned to each reason **shall** be used to represent the reason in SIS Access Protocol (Annex S).

A.2.2.5 S_UNBIND_INDICATION Primitive

Name :
S_UNBIND_INDICATION ( )
Arguments :
1. Reason
Direction :
Subnetwork Interface->Client
Description :
The S_UNBIND_INDICATION primitive **shall** (1) be issued by the Subnetwork

Interface Sublayer to unilaterally declare a client as off-line. If the client wants to come on-line again, it must issue a new a S_BIND_REQUEST primitive as specified in Section A.2.1.1.

The S_UNBIND_INDICATION primitive provides a means for the Subnetwork Interface Sublayer to manage the clients connected to it.

The *Reason* argument of the S_UNBIND_INDICATION primitive **shall** (2) specify why the client was declared off-line.

Reason	Value
Reserved	1
Inactivity (failure to respond to “Keep alive”)	2
Too many invalid primitives	3
Reserved	4
ARQ Mode Unsupportable during	5
SIS Service Terminating	6

The value assigned to each reason **shall** be used to represent the reason in SIS Access Protocol (Annex S).

A.2.2.6 S_UNIDATA_REQUEST Primitive

Name :

S_UNIDATA_REQUEST( )

Arguments :

1. Priority
2. Destination SAP ID
3. Destination Node Address
4. Delivery Mode (Service Type)
5. TimeToLive (TTL)
6. Size of U_PDU
7. U_PDU (User Protocol Data Unit)

Direction :

Client->Subnet Interface

Description :

The S_UNIDATA_REQUEST primitive **shall** (1) be used by connected clients to submit a U_PDU to the HF subnetwork for delivery to a receiving client.

The argument *Priority* **shall** (2) represent the priority of the U_PDU. The U_PDU priority **shall** (5) take a value in the range 0-15, with 0 being the lowest priority and 15 the highest. The processing by HF protocol sublayers **shall** (6) make a “best effort” to give precedence to high priority U_PDUs over lower priority U_PDUs which are queued in the system.

The argument *Destination SAP ID* **shall** (3) specify the SAP ID of the receiving client. Note that as all nodes will have uniquely specified SAP IDs for clients, the Destination SAP ID distinguishes the destination client from the other clients bound to the destination node.

The argument *Destination Node Address* **shall** (4) specify the HF subnetwork address of the physical HF node to which the receiving client is bound.

The argument *Delivery Mode* (Service Type) **shall** (5) be comprised of the four elements specified here. This argument can be given the value of "DEFAULT" which means that the delivery mode associated with the U_PDU will be the Default Service Type specified by the client during "binding" (i.e., the value DEFAULT is equal to the *Service Type* argument of client's original S_BIND_REQUEST). Values other than DEFAULT for the *Delivery Mode* can be used to override these default values.

Service Type comprises the following elements:

1. *Transmission Mode for the Service.* --- ARQ or Non-ARQ Transmission Mode **shall**(6) be specified, with one of the Non-ARQ submodes if Non-ARQ was requested. Non-ARQ transmission has two submodes such as: *Error-Free-Only* delivery to destination client (default), and *Delivery-with-Errors* which provides delivery to destination client even with *some* errors. The *Delivery-with-Errors* is optional and use of this value is associated with the optional NON-ARQ WITH ERRORS profile option.

2. *Data Delivery Confirmation for the Service* --- The client **may** request none, one or both of the Data Delivery Confirmation modes for the service. There are two types of data delivery confirmation:
    - Node-to-Node Delivery Confirmation
    - Client-to-Client Delivery Confirmation

    The client can request explicit confirmation, i.e, Node-to-Node or Client-to-Client, from the Subnetwork to provide indication that its U_PDUs have been properly delivered to their destination. Explicit delivery confirmation **shall** (9) be requested only in combination with ARQ delivery.

    [Note: The Node-to-Node Delivery Confirmation does not require any explicit peer-to-peer communication between the Subnetwork Interface Sublayers and hence it does not introduce extra overhead. It simply uses the ACK (ARQ) confirmation provided by the Data Transfer Sublayer. Client-to-Client Delivery Confirmation requires explicit peer-to-peer communication between the Sublayers and therefore introduces overhead. It should be used only when it is absolutely critical for the client to know whether or not its data was delivered to the destination client (which may, for instance, be disconnected).]

    NOTE: This service definition allows for both types of delivery confirmation to be requested. Annex S (Edition 4) only allows for one type to be requested.

3. *Order of delivery of any U_PDU to the receiving client.* --- A client **shall** (10) request that its U_PDUs are delivered to the destination client "in-order" (as they are submitted) or in the order they are received by the destination node.

    "in order" **shall not** be requested for the Non-ARQ service.

    NOTE: Use of "in-order" is implemented in the DTS and this leads to interaction

between different clients using “in-order” and can lead to unnecessary data loss. It is recommended to avoid use of “in-order” and to handle ordering within the client application.

4. *Minimum Number of Retransmissions* --- This argument **shall** (12) be valid if and only if the Transmission Mode is a Non-ARQ type. If the Transmission Mode is a Non-ARQ type, then the subnetwork **shall** (13) retransmit each U_PDU the number of times specified by this service. If this is not specified, the U_PDU is sent only once.
   [Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]

The argument *TimeToLive (TTL)* **shall** (6) specify the maximum amount of time the submitted U_PDU is allowed to stay in the HF Subnetwork before it is delivered to its destination. If the TTL is exceeded the U_PDU **shall** (7) be discarded. A TTL value of 0 **shall** (8) define an infinite TTL**,** i.e. the subnetwork should try *forever* to deliver the U_PDU.

The subnetwork **shall** (9) have a default maximum TTL. The default maximum TTL **shall**(10) be configurable as an implementation-dependent value. As soon as the Subnetwork Interface Sublayer accepts a S_UNIDATA_REQUEST primitive, it **shall** (11) immediately calculate its *TimeToDie (TTD)* by adding the specified TTL (or the default maximum value if the specified TTL is equal to 0) to the current Time of Day. The TTD attribute of a U_PDU **shall** (12) accompany it during its transit within the subnetwork. [Note that the TTD is an absolute time while the TTL is a time interval relative to the instant of the U_PDU submission.]

The *Size of U_PDU* argument **shall** (13) be the size of the U_PDU that is included in this S_UNIDATA_REQUEST Primitive.

The final argument, *U_PDU,* **shall** (14) be the actual Data Unit submitted by the client to the HF Subnetwork.

A.2.2.7 S_UNIDATA_REQUEST_CONFIRM Primitive

Name :
S_UNIDATA_REQUEST_CONFIRM
Arguments :
1. Destination Node Address
2. Destination SAP ID
3. Size of Confirmed U_PDU
4. U_PDU (User Protocol Data Unit or part of it)

Direction :
Subnetwork Interface->Client

Description :
The S_UNIDATA_REQUEST_CONFIRM primitive **shall** (1) be issued by the Subnetwork Interface Sublayer to acknowledge the successful delivery of a S_UNIDATA_REQUEST submitted by the client.

This primitive **shall** (2) be issued only if the client has requested Data Delivery Confirmation (either during binding or for this particular data unit).

The *Destination Node Address* argument in the S_UNIDATA_REQUEST_CONFIRM Primitive **shall** have the same meaning and be equal in value to the *Destination Node Address* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_ CONFIRM Primitive is the response.

The *Destination SAP_ID* argument in the S_UNIDATA_REQUEST_ CONFIRM Primitive **shall** (4) have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_CONFIRM Primitive is the response.

The *Size of Confirmed U_PDU* argument **shall** (5) be the size of the U_PDU or part that is included in this S_UNIDATA_REQUEST_CONFIRM Primitive.

The *U_PDU* argument in the S_UNIDATA_REQUEST_CONFIRM Primitive **shall** (6) be a copy of the whole or a fragment of the *U_PDU* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_CONFIRM Primitive is the response.

Using these arguments, the client will usually be able to uniquely identify the U_PDU that is being acknowledged. Depending on the implementation of the protocol, the last argument, *U_PDU,* may not be a complete copy of the original U_PDU but only a partial copy, i.e., only the first X bytes are copied for some value of *X*. If a partial U_PDU is returned, *U_PDU_response_frag_size* bytes **shall** (9) be returned to the client starting with the first byte of the U_PDU so that the client will have the U_PDU segment information. The number of bytes returned, *U_PDU_response_frag_size*, **shall**(10) be a configurable parameter in the implementation..

A.2.2.8 S_UNIDATA_REQUEST_REJECTED Primitive

Name :

S_UNIDATA_REQUEST_REJECTED

Arguments :

1. Reason
2. Destination Node Address
3. Destination SAP ID
4. Size of Rejected U_PDU (or part)
5. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

The S_UNIDATA_REQUEST_REJECTED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer to inform a client that a S_UNIDATA_REQUEST was not delivered successfully.

This primitive **shall** (2) be issued if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU) and the data was unsuccessfully delivered. This primitive also **shall** (3) be issued to a client if a U_PDU larger than the MTU is submitted.

The argument *Reason* **shall** (4) specify why the delivery failed, using the encoding given in the table below:

Reason	Value
TTL Expired	1
Destination SAP ID not bound	2
Destination node not responding	3
U_PDU larger than MTU	4
Tx Mode not specified	5
Broadcast Not Allowed	6
Address Not Known	7
No ALE Mapping for Address	8
Soft Link Terminated for higher priority link	9
Soft Link Terminated to share channel	10
Idle Soft Link Terminated	11
Rejected due to EMCON state	12
Data Buffers Full	13
Address not Routable	14
Other	15

The value assigned to each reason **shall** be used to represent the reason in SIS Access Protocol (Annex S).

The *Destination Node Address* argument in the S_UNIDATA_REQUEST_REJECTED Primitive **shall**
have the same meaning and be equal in value to the *Destination Node Address* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Destination SAP_ID* argument in the S_UNIDATA_REQUEST_REJECTED Primitive **shall** (7) have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Size of Rejected U_PDU* argument **shall** (8) be the size of the U_PDU or part that is included in this S_UNIDATA_REQUEST_REJECTED Primitive.

Just as specified for the S_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* argument in the S_UNIDATA_REQUEST_REJECTED primitive may only be a partial

copy of the original U_PDU, depending on the implementation of the protocol. If a partial U_PDU is returned, *U_PDU_response_frag_size* bytes **shall** (9) be returned to the client starting with the first byte of the U_PDU so that the client will have the U_PDU segment information. The number of bytes returned, *U_PDU_response_frag_size*, **shall**(10) be a configurable parameter in the implementation.

A.2.2.9 S_UNIDATA_INDICATION Primitive

Name :

S_UNIDATA_INDICATION

Arguments :

1. Priority
2. Destination SAP ID
3. Destination Node Address
4. Transmission Mode
5. Source SAP ID
6. Source Node Address
7. Size of U_PDU
8. Number of Blocks in Error
9. Array of Block-Error Pointers
10. Number of Non-Received Blocks
11. Array of Non-Received-Block Pointers
12. U_PDU

Direction :

Subnetwork Interface->client

Description :

The S_UNIDATA_INDICATION primitive **shall** (1) be used by the Subnetwork Interface Sublayer to deliver a received U_PDU to the client.

The *Priority* argument **shall** (2) be the priority of the U_PDU.

The *Destination SAP ID* argument **shall** (3) be the SAP ID of the client to which this primitive is delivered.

The *Destination Node Address* argument **shall** (4) be the address assigned by the sending node to the U_PDU contained within this primitive. This normally will be the address of the local (i.e., receiving) node. It may however be a “group” address to which the local node has subscribed (Group Addresses and their subscribers are defined during configuration) and to which the source node addressed the U_PDU.

The *Transmission Mode* argument **shall** (5) be the mode by which the U_PDU was transmitted by the remote node and received by the local node; ie, ARQ, Non-ARQ (Broadcast) transmission, or Non-ARQ w/ Errors.

The *Source SAP ID* **shall** (6) be SAP ID of the client that sent the U_PDU.

The *Source Node Address* **shall** (7) represent the node address of the client that sent the U_PDU.

The *Size of U_PDU* argument **shall** (8) be the size of the U_PDU that was sent and delivered in this S_UNIDATA_INDICATION S_Primitive.

The following four arguments **shall** (9) be present in the S_UNIDATA_INDICATION S_Primitive if and only if the Transmission Mode for the U_PDU is equal to Non-ARQ w/ Errors. Note that this service is optional and only supported as part of the optional NON-ARQ WITH ERRORS profile option:

a) The *Number of Blocks in Error* argument **shall** (10) equal the number of data blocks in the U_PDU that were received in error by the lower layers of the subnetwork and that were passed on to the Subnetwork Interface Sublayer. This argument **shall** (11) specify the number of ordered pairs in the *Array of Block-Error Pointers* argument.

b) The *Array of Block-Error Pointers* argument **shall** (12) consist of a an array of ordered pairs, the first element in the pair equal to the location within the U_PDU of the data block with errors, and the second element equal to the size of the data block with errors.

c) The *Number of Non-Received Blocks* argument **shall** (13) equal the number of data blocks missing from the U_PDU because they were not received. This argument **shall** (14) specify the number of ordered pairs in the *Array of Non-Received-Block Pointers* argument.

d) The *Array of Non-Received-Block Pointers* **shall** (15) consist of an array of ordered pairs, the first element in the pair equal to the location of the missing data block in the U_PDU and the second element equal to the size of the missing data block.

The final argument, *U_PDU*, **shall** (16) contain the actual received user data for delivery to the client.

A.3 Peer-to-Peer Communication Protocols and S_PDUs

Peer Subnetwork Interface Sublayers, generally in different nodes, **shall** (1) communicate with each other by the exchange of Subnetwork Interface Sublayer Protocol Data Units (S_PDUs).

For the Subnetwork configurations currently defined in STANAG 5066, Peer-to-Peer Communication **shall** be (2)
required for the Exchange of Client Data, including confirmation.

Explicit Peer-to-Peer communication **shall** (3) not be required for the establishment or termination of Soft Link or Broadcast Data Exchange sessions.

The Peer-to-Peer communication required for the exchange of Client Data is similar for all Data exchange sessions, using the facilities of lower sublayers in the protocol profile. The encoding of the S_PDUs and the protocol governing the Peer-to-Peer Communication are described in the following sections.

A.3.1 Subnetwork Interface Sublayer Protocol Data Units (S_PDUS) and Encoding Requirements

There are currently three types of S_PDUs. Additional S_PDU types may be defined in the future. The generic encoding of the eight S_PDU types showing the fields and subfields of the S_PDUs is shown in Figure A-1.

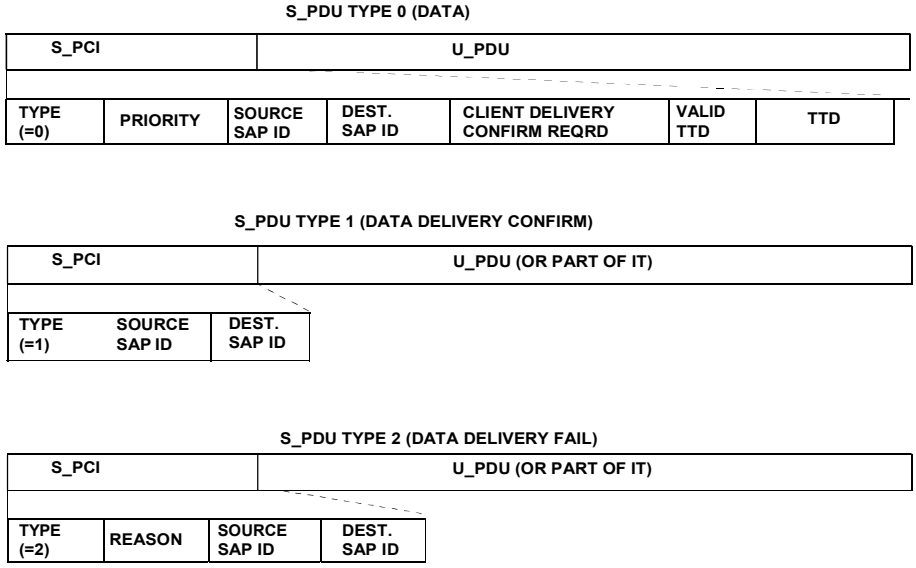


Figure A-1: Generic Encoding of S_PDUs

The first encoded field **shall** (1) be common to all S_PDUs. It is called “TYPE” and **shall** (2) encode the type value of the S_PDU as follows:

S_PDU TYPE	S_PDU Name
0	DATA
1	DATA DELIVERY CONFIRM
2	DATA DELIVERY FAIL
3-7	RESERVED

The meaning and encoding of the remaining fields, if any, in an S_PDU **shall** (3) be as specified in the subsection below corresponding to the S_PDU type. Values 3-7 were used in Edition 3 and are noted as reserved in this edition.

A.3.1.1 DATA S_PDU

Type :

"0" = DATA S_PDU

Encoding :

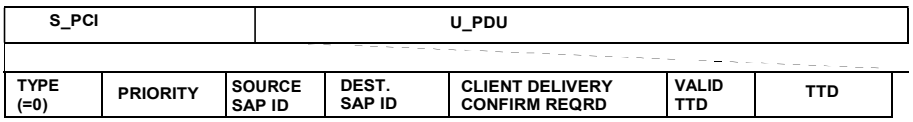


Figure A-2: Generic Encoding of the DATA S_PDU

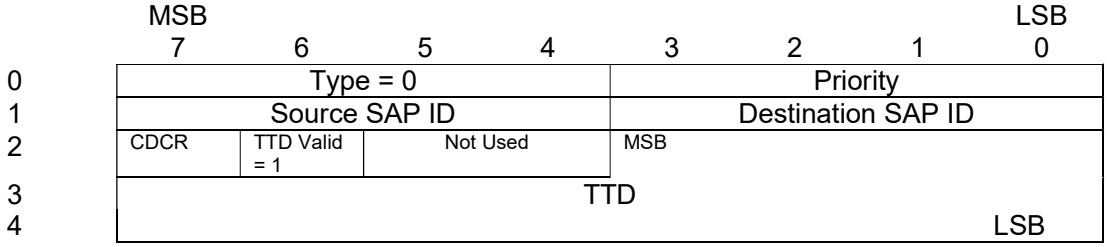


Figure A-3: Bit-Field Map of the DATA S_PDU S_PCI with valid TTD

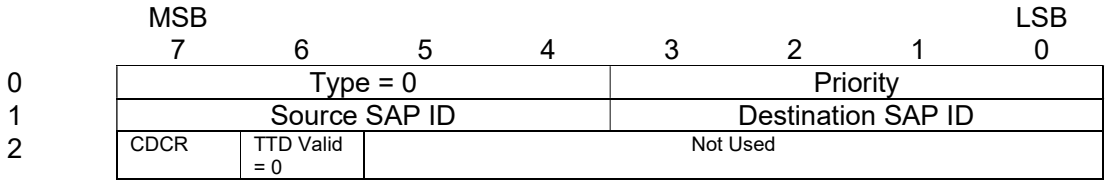


Figure A-4: Bit-Field Map of the DATA S_PDU S_PCI without TTD

Description :

The DATA S_PDU **shall** (1) be transmitted by the Subnetwork Interface Sublayer in order to send client data to a remote peer sublayer.

The DATA S_PDU shall (2) be encoded as specified in Figure A-2, Figure A-3, Figure A-4 and in the paragraphs below.

This S_PDU **shall** (3) consist of two parts:

a) the first part **shall** (4) be the S_PCI (Subnetwork Interface Sublayer Protocol Control Information) and represents the overhead added by the sublayer;

b) the second part **shall** (5) be the actual client data (U_PDU).

The first field of four bits the S_PCI part **shall** (6) be “TYPE”. Its value **shall** (7) be equal to 0 and identifies the S_PDU as being of type DATA.

The second field of four bits **shall** (8) be “PRIORITY” and represents the priority of the client’s U_PDU. The “PRIORITY” field **shall** (9) be equal in value to the corresponding argument of the S_UNIDATA_REQUEST primitive submitted by the client..

The third field of four bits of the S_PCI **shall** (10) be the “SOURCE SAP ID” and identifies the client of the transmitting peer which sent the data.

The fourth field of four bits **shall** (11) be the “DESTINATION SAP ID” and identifies the client of the receiving peer which must take delivery of the data. There is no need to encode the source and destination node addresses in the S_PDU as this information is relayed between the peers by the underlying sublayers. The “DESTINATION SAP ID” **shall** (12) be equal in value to the corresponding argument of the S_UNIDATA_REQUEST primitive submitted by the client

The fifth field of the S_PCI **shall** (13) be “CLIENT DELIVERY CONFIRM REQUIRED”, and is encoded as a single bit that can take the values “YES” (=1) or “NO” (=0). The value of this bit **shall** (1 be set according to the *Delivery Mode* requested explicitly for this U_PDU (see S_UNIDATA_REQUEST Primitive), which **may** be defaulted to the *Service Type* requested by the sending client during binding (see S_BIND_REQUEST primitive).

The sixth field **shall** (15) be the VALID TTD field, and is encoded as a single bit that can take the values “YES” (=1) or “NO” (=0), indicating the presence of a valid TTD within the S_PCI.

The seven field of the S_PCI **shall** (14) be two unused bits that are reserved for future use.

The eighth and last field of the S_PCI **shall** (15) be “TTD” and represents the TimeToDie for this U_PDU. The first four bits of this field **shall** (16) have meaning if and only if the VALID TTD field equals “YES”, the remaining 16 bits of the field **shall** (17) be present in the S_PCI if and only if the VALID TTD field equals “YES”.

The TTD field encodes the Julian date modulo 16, and the GMT in seconds after which time the S_PDU must be discarded if it has not yet been delivered to the client. The simple Julian date system, which numbers the days of the year consecutively starting with 001 on 1 January and ending with 365 on 31 December (or 366 on leap years). The Julian date modulo 16 part of the TTD **shall** be mapped into the first four bits of the TTD field (i.e., bits 0-3 of byte 2 of the S_PDU).

The 16 high bits of the GMT part of the TTD shall be mapped into the 2 remaining bytes of the TTD field; the LSB of the GMT shall be discarded. If the “VALID TTD” flag bit of a DATA S_PDU is set (=1) then the complete TTD 20-bit field is present and its value must be used. If this flag bit is not set (=0), the last two bytes of the TTD field are not present (to conserve overhead) and the TTD must not be used. The “VALID TTD” flag bit allows the transmitting peer to specify whether the receiving peer should discard the S_PDU by based on TTD or it delivered the U_PDU to the client without consideration of the TTD.

A.3.1.2 DATA DELIVERY CONFIRM S_PDU

Type :

"1" = DATA DELIVERY CONFIRM

Encoding :

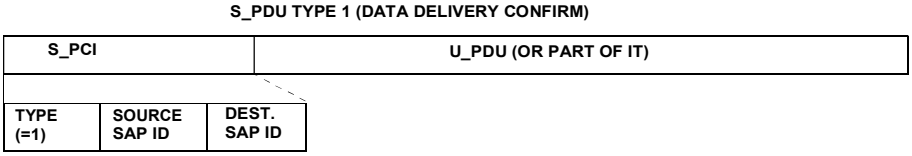


Figure A-5: Generic Encoding of the DATA DELIVERY CONFIRM S_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type = 1				Priority			
1	Source SAP ID				Destination SAP ID			

Figure A-6: Bit-Field Map of the DATA DELIVERY CONFIRM S_PDU S_PCI

Description :

The DATA DELIVERY CONFIRM S_PDU **shall** (1) transmitted in response to a successful delivery to a Client of a U_PDU which was received in a DATA type S_PDU in which the "CLIENT DELIVERY CONFIRM REQUIRED" field was set to "YES". The DATA DELIVERY CONFIRM
S_PDU **shall** be (2) transmitted by the Subnetwork Interface Sublayer to the peer sublayer which originated the DATA type S_PDU.

The first part of the DATA DELIVERY CONFIRM S_PDU **shall** (3) be the S_PCI, while the second part
shall (4) be a full or partial copy of the U_PDU that was received and delivered to the destination Client.

The first field of the S_PCI part **shall** (5) be "TYPE" and its value **shall** (6) equal 1 to identify the S_PDU as being of type DATA DELIVERY CONFIRM.

The remaining fields and their values for the S_PCI part of the DATA DELIVERY CONFIRM S_PDU **shall** (7) be equal in value to the corresponding fields of the DATA S_PDU for which this DATA DELIVERY CONFIRM S_PDU is a response.

The peer sublayer that receives the DATA DELIVERY CONFIRM **shall** (8) inform the client which originated the U_PDU that its data has been successfully delivered to its Destination by issuing a S_UNIDATA_REQUEST_CONFIRM in accordance with the data exchange protocol of Section A.3.2.3.

A.3.1.3 DATA DELIVERY FAIL S_PDU

Type :

"2" = DATA DELIVERY FAIL

Encoding :

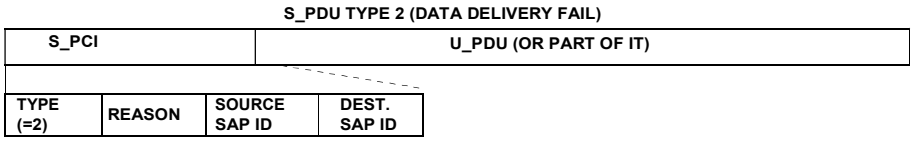


Figure A-7: Generic Encoding of the DATA DELIVERY FAIL S_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type = 2				Priority			
1	Source SAP ID				Destination SAP ID			

Figure A-8: Bit-Field Map of the DATA DELIVERY FAIL S_PDU S_PCI

Description :

The DATA DELIVERY FAIL S_PDU **shall** (1) be transmitted in response to a failed delivery to a Client of a U_PDU that was received in a DATA type S_PDU with the "CLIENT DELIVERY CONFIRM REQUIRED" field set to "YES".

The first part of this S_PDU **shall** (2) be the S_PCI.

The second part **shall** (3) be a full or partial copy of the U_PDU that was received but not delivered to the destination Client.

The first field of the S_PCI **shall** (4) be "TYPE". Its value **shall** (5) be equal to 2 and identifies the S_PDU as being of type DATA DELIVERY FAIL.

The second field **shall** (6) be "REASON" and explains why the U_PDU failed to be delivered. It can take a value in the range 0-15; valid reasons are defined in the table below.

Reason	Value
Unassigned and reserved	0
Destination SAP ID not bound	1
Unassigned and reserved	2-15

The SOURCE SAP_ID and DESTINATION SAP_ID fields of the S_PCI **shall** (7) be equal in value to the corresponding fields of the DATA S_PDU for which the DATA DELIVERY FAIL S_PDU is a response.

The peer sublayer that receives the DATA DELIVERY FAIL S_PDU, **shall** (8) inform the client which originated the U_PDU that its data was not delivered to the destination by

issuing a S_UNIDATA_REQUEST_REJECTED primitive, in accordance with the data exchange protocol of Section A.3.2.3.

A.3.2 Peer-to-Peer Communication Protocol

This section specifies the protocols governing the Peer-to-Peer communication for Establishing and Terminating Soft Link Data Exchange Sessions, Establishing and Terminating Broadcast Data Exchange Sessions and Exchanging Client Data. In these specifications, the node whose local client or Subnetwork Interface Sublayer first requests a Data Exchange Session is denoted as the caller or calling node and the remote node is denoted as the called node.

A.3.2.1 Soft Link Data Exchange Session

The Subnetwork Interface Sublayer initiates Soft Link Data Exchange Sessions with remote peers based on the destinations of queued client U_PDUs. In particular, sublayer management algorithms must be established to initiate the protocols for establishment or termination a Soft Link Data Exchange Session. This STANAG allows these sublayer management algorithms to be based on implementation dependent criteria and factors. The use of comparative U_PDU queue-length for given clients, source-destination sets and priority levels for any implementation is allowed and expected (even if the algorithms are trivial) but remain beyond the scope of this STANAG.

A.3.2.1.1 Soft Links for Non-ARQ data

Soft links are always established prior to transfer of ARQ data over the CAS.

When CAS is used in Multiple Simultaneous Peer Access or ALE 1:1 EXPLICIT CAS-1 mode, as specified in Annex B Section B.5, Non-ARQ will always be transferred without a Soft link.

When CAS is used in ALE 1:1 IMPLICIT CAS-1 mode, as specified in Annex B Section B.6, a soft link **shall** be established for all data. The CAS signals this to the SIS by rejecting Non-ARQ data with reason “Physical Link Needed”. The SIS layer **shall** establish a soft link and then resend the non-ARQ data. Note that this will only occur when interoperating with an Ed4 peer, as this mode was not specified in Edition 3.

A.3.2.1.2 Protocol for Establishing a Soft Link Data Exchange Session

The establishment of Soft Link Data Exchange Sessions **shall** (1) not require explicit peer-to-peer handshaking within the Subnetwork Interface Sublayer.

The calling peer **shall** (2) implicitly establish a Soft Link Data Exchange Session by requesting its Channel Access Sublayer to make a physical link to the required

remote node, using the procedure for making physical links specified in Annex B. In accordance with these procedures, both peer Subnetwork Interface Sublayers (i.e., the calling and called sublayers) are informed about the successful making of a physical link between their nodes by their respective Channel Access Sublayers.

After the physical link is made, both peer Subnetwork Interface Sublayers **shall** (3) declare that the Soft Link Data Exchange Session has been established between the respective source and destination nodes. Data may then be exchanged in accordance with the protocols specified in Section A.3.2.4.

A.3.2.1.3 **Protocol for Terminating a Soft Link Data Exchange Session**

No peer-to-peer communication by the Subnetwork Interface Sublayer **shall** (1) be required to terminate a Soft Link Data Exchange Session.

A Soft Link Data Exchange Session **shall** (2) be terminated by either of the two peers by a request to its respective Channel Access Sublayer to break the Physical Link in accordance with the procedure specified in Annex B. Both Subnetwork Interface sublayers will be informed about the breaking of the Physical link by their respective Channel Access Sublayers.

Since a called peer can terminate a Soft Link Data Exchange Session if it has higher priority data destined for a different Node, called peers **shall** (3) wait a configurable minimum time before unilaterally terminating sessions, to prevent unstable operation of the protocol.

> Note: The caller sublayer normally initiates the termination of the session (by breaking the physical link) based on the destinations of its queued U_PDUs, and on any ongoing communication with the distant node. The inter-layer signaling for coordination would normally be carried out via the subnetwork management sublayer. The called sublayer can also terminate the session if it has high priority data destined for a different node. However, called sublayers should wait a configurable minimum time before unilaterally terminating sessions, otherwise an unstable condition may arise if all nodes in the network have data to transmit and called sublayers immediately close sessions in order to establish other sessions as callers. If such a situation arises, the efficiency of a subnetwork will deteriorate as a result of nodes continuously establishing and terminating sessions without actually transmitting data. The minimum amount of time that a called sublayer should wait before it attempts to terminate a Soft Link Session must be carefully chosen and will depend on a number of factors such as the subnetwork size and configuration. Specification of this and other parameters as a configurable but required value allows implementations of the STANAG to be tuned for specific network, with the values for these parameters distributed as part of the standard operating procedures for a given network.

After the Subnetwork Interface Sublayer has been notified that the Physical Link has been broken, the Subnetwork Interface Sublayer **shall** (4) declare the Soft Link Exchange Session as terminated.

A.3.2.1.4 Queue Management

The intent of this STANAG, as noted above, is to maximize flexibility of implementation when managing queues.

When the CAS follows the Multiple Simultaneous Peer Access model described in Annex B, the SIS layer will be able to establish multiple queues to different peers, each with an open soft link. In this model the DTS level queues will handle D_PDUs of different priority according to priority. Once a soft link is open, the SIS **may** pass all data to the DTS and allow DTS to handle priority and sharing of resource to transmit data to different peers. Annex C, section 7.5.2 describes DTS handling of priority and peer sharing.

When the Single Peer Access model is used in a deployment with multiple peers queueing is done at the SIS level, except in ALE 1:1 EXPLICIT CAS-1 mode. This is because a limited number of soft links can be open at a time, and the SIS will need to queue data for peers where the soft link is not open. This will commonly just be one soft link, but can be more when the STANAG 5066 server is connected to multiple physical channels. The SIS will need to control which links are open. The following rules apply:

- If a soft link is open and traffic of a higher priority than any being transmitted arrives for a different peer, the SIS **shall** close active link as quickly as possible, noting the considerations in Section A.3.2.1.3.
- Where there is traffic for multiple peers, the SIS **shall** ensure fairness between the peers and open soft links to each peer in turn at reasonable intervals. Traffic to one peer **shall not** be allowed to prevent traffic to other peers.

The SIS **shall** close a soft link using the CAS C_PHYSICAL_LINK_BREAK primitive with reason "Soft Link Terminated for higher priority link" or "Soft Link Terminated to share channel" or "Idle Soft Link Terminated".

Data rejected by the DTS will return D_UNIDATA_REQUEST_REJECTED to the CAS, which will be passed to the SIS as C_UNIDATA_REQUEST_REJECTED. When this is due to soft link termination, the reason for the reject **shall** match the reason given by SIS for closing the soft link. The SIS will need to make a balance between rapid switching of soft link and allowing transfers on soft link to complete before closing the soft link. This is an implementation choice.

If Annex B Multicast ALE mode is supported, this needs dedicated use of the channel. For this reason, all soft links will need to be closed before multicast or broadcast data can be sent. Queue management needs to take this into account.

A.3.2.2 Protocol for Establishing and Terminating a Broadcast Data Exchange Session

No explicit peer-to-peer communication **shall** (1) be required to establish and terminate a Broadcast Data Exchange Session. A Broadcast Data Exchange Session is established and terminated either by a management process or unilaterally by the Subnetwork Interface Sublayer based on a number of criteria as explained in section A.1.1.3.

As noted in section A.1.1, clients may interleave requests for data-exchange sessions. At some point, the subnetwork might also be configured to provide exclusive support for a Broadcast Data Exchange Session. In this case, when the subnetwork is first configured by the local (implementation-dependent) management function to provide exclusive support for a Broadcast Data Exchange Session the Subnetwork Interface Sublayer **shall** (2) send an S_UNBIND_INDICATION to any bound clients that had requested ARQ Delivery Service, with the REASON = “ARQ Mode Unsupportable during Broadcast Session”. Subsequent S_BIND requests by clients requesting ARQ service **shall** (3) be rejected with the same reason.

A.3.2.3 Protocol for Exchanging Client Data

After a Data Exchange Session of any type has been established, sublayers with client data to exchange **shall** (1) exchange DATA (TYPE 0) S_PDUs using the protocol specified below and in accordance with the service characteristics of the respective session.

The sublayer **shall** (2) discard any U_PDU submitted by a client where the U_PDU is greater in size than the Maximum Transmission Unit (MTU) size assigned to the client by the S_BIND_ACCEPTED Primitive issued during the client-bind protocol.

If a U_PDU is discarded because it exceeded the MTU size limit and if the DELIVERY CONFIRMATION field for the U_PDU specifies CLIENT DELIVERY CONFIRM or NODE DELIVERY CONFIRM, the sublayer **shall** (3) notify the client that submitted the U_PDU as follows:

- the sublayer **shall** send a S_UNIDATA_REQUEST_REJECTED Primitive to the client;
- the REASON field **shall** be equal to “U_PDU Larger than MTU”.

For U_PDUs that have been accepted for transmission, the sending sublayer retrieves client U_PDUs and their associated implementation-dependent service attributes (such as the S_Primitive that encapsulated the U_PDU) from its queues (according to Priority and other implementation-dependent criteria), and proceeds as follows:

- the sending sublayer **shall** (7) encode the retrieved U_PDU into a DATA (TYPE 0) S_PDU, transferring any service attributes associated with U_PDU to the S_PDU as required;

- the sending sublayer **shall** (8) encode the resulting DATA (TYPE 0) S_PDU in accordance with the C_Primitive interface requirements of the Channel Access Sublayer as specified in Annex B, i.e,:

- the sublayer **shall** (9) encode the S_PDU as a C_UNIDATA_REQUEST Primitive of the priority corresponding to that initially specified by the client in the S_Primitive, otherwise;

- the sending sublayer then **shall** (11) pass the resulting C_primitive to the Channel Access Sublayer for further processing to send the DATA (TYPE 0) S_PDU to its remote peer.

- if the service attributes for the U_PDU require NODE DELIVERY CONFIRMATION, the sublayer **shall** (12) wait for a configurable time for a response as follows:

- if the sublayer receives a C_UNIDATA_REQUEST_CONFIRM prior to the end of the waiting time, the sublayer **shall** (13) send to the client a S_UNIDATA_REQUEST_CONFIRM Primitive;

- otherwise, if the sublayer receives a C_UNIDATA_REQUEST_REJECTED the sublayer **shall** (14) send to the client a S_UNIDATA_REQUEST_REJECTED Primitive, unless the reject reason is “Physical Link Needed”. If the rejection is for this reason (which is expected only for Non-ARQ data), the sublayer **shall** establish a soft link and resubmit the C_UNIDATA_REQUEST;

- otherwise, if the waiting time ends prior to receipt of any response indication from the Channel Access sublayer, the Subnetwork Interface sublayer **shall** (15)
  send to the client a S_UNIDATA_REQUEST_REJECTED Primitive. The REASON field shall be set equal to “Destination Node Not Responding”.

- if the service attributes for the U_PDU require CLIENT DELIVERY CONFIRMATION, the sending sublayer shall (16) wait for a configurable time for a response as follows:

- if the Subnetwork Interface sublayer receives a C_Primitive confirming node-node delivery (i.e., a C_UNIDATA_REQUEST_CONFIRM Primitive) and a “DATA DELIVERY CONFIRM” (TYPE 1) S_PDU is received from the remote sublayer prior to the end of the waiting time, the Subnetwork Interface sublayer **shall** (17) send to the client a

S_UNIDATA_REQUEST_CONFIRM Primitive;

- otherwise, if the Subnetwork Interface sublayer receives either a “reject” C_Primitive from the Channel Access Sublayer or a “DATA DELIVERY FAIL” (TYPE 2) S_PDU from the remote peer prior to the end of the waiting time, the Subnetwork Interface sublayer **shall** (18) send to the client either a S_UNIDATA_REQUEST_REJECTED Primitive. The REASON field **shall** (18.1) be derived from the “DATA DELIVERY FAIL” (TYPE 2) S_PDU or the reject C_Primitive that was received;

- otherwise, if the waiting time ends prior to receipt of a response message, the sublayer **shall** (19) send to the client a S_UNIDATA_REQUEST_REJECTED Primitive. The REASON field shall be set equal to “Destination Node Not Responding”.

- On completion of these actions by the sending sublayer the client data delivery protocol terminates for the given DATA (TYPE 0) S_PDU.

A receiving sublayer manages the client data exchange protocol as follows:

- the receiving sublayer **shall** (20) accept encoded DATA (TYPE 0) S_PDUs from the Channel Access Sublayer using C_Primitives in accordance with the interface requirements specified in Annex B.

- the receiving sublayer **shall** (21) extract the U_PDU, Destination SAP_ID and the other associated service attributes from the DATA (TYPE 0) S_PDUs as required;

- if there is no client bound to the destination SAP_ID, the receiving sublayer **shall** (22) discard the U_PDU by; otherwise,

- the sublayer shall (23) deliver the extracted U_PDU to the destination client bound to Destination SAP_ID using a S_UNIDATA_INDICATION Primitive;

- if the received S_PDU has the “CLIENT DELIVERY CONFIRM REQUIRED” field set equal to “YES”, then the sublayer **shall** (25) provide delivery confirmation as follows:

  - if a client was bound to the Destination SAP_ID, the sublayer **shall** (26) encode as required and send a “DATA DELIVERY CONFIRM” (TYPE 1) S_PDU to the sending sublayer;

  - if a client was not bound to the Destination SAP_ID, the sublayer **shall** (27) encode as required and send a “DATA DELIVERY FAIL” (TYPE 2)

S_PDU to the sending sublayer.

- On completion of these actions by the receiving sublayer the client data delivery protocol terminates for the given DATA (TYPE 0) S_PDU.

Implementation-dependent queuing disciplines, flow-control procedures, or other characteristics in the sublayer **shall** (28) not preclude the possibility of managing the data exchange protocol for more than one U_PDU at a time. In particular, the Subnetwork Interface Sublayer **shall** (29) be capable of sending a U_PDU, encapsulated in a DATA (TYPE 0) S_PDU and C_Primitive as required, to the Channel Access Sublayer prior to receipt of the data-delivery-confirm response for a U_PDU sent earlier.

[Note: This requirement mitigates the reduction in link throughput that occurs when a subnetwork ceases transmission of any U_PDUs while it awaits confirmation of their delivery. The performance degradation is typical of that which occurs when using a STOP-AND-WAIT form of ARQ protocol anywhere in a communication system.]

The nominal procedures for exchanging DATA S_PDUs for both the Sending and Receiving Peers are shown in Figure A-9 and Figure A-10. This STANAG acknowledges that other implementations may satisfy the requirements stated above.

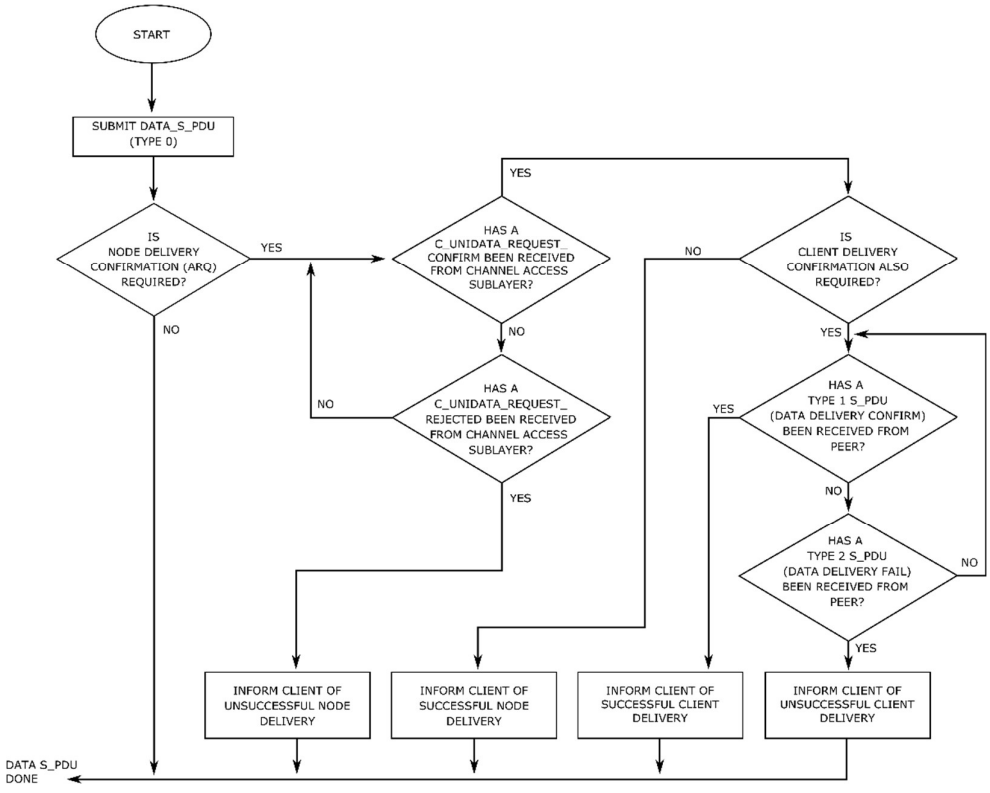


Figure A-9: Data Exchange Procedures: SENDING PEER

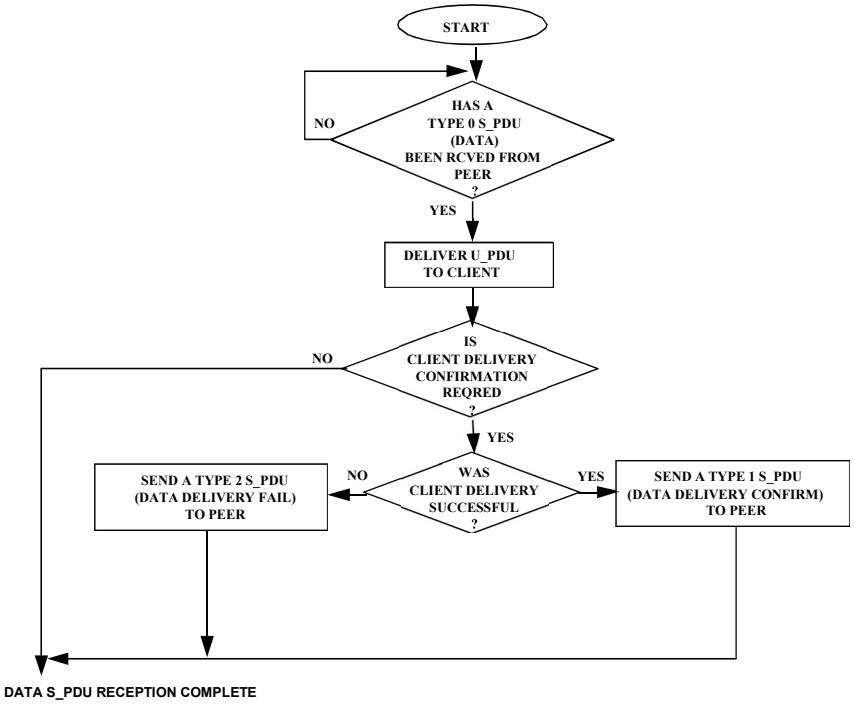


Figure A-10: Data Exchange Procedures: RECEIVING PEER

A.4 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.

A.4.1 **Hard Links**

This optional deprecated service is defined as the HARD LINKS profile option.

Hard Links are specified in Edition 3 Annex A in a manner that does not conflict with any of the Edition 4 specification. So, use of the Edition 3 specification is straightforward. Key sections:

- Sections A.1.1.2: overview of the service
- Table A-1: summary of the hard link services
- Sections A.2.1.18 – A.2.1.25: detailed definition of the hard link services
- Section A.3.1 and Sections A.3.1.4 – A.3.1.7: encoding of hard links in the peer SIS protocol

- Section A.3.2.2: procedures of operation
- Use of Rank, as specified in Edition 3, is also used to control details of the hard link protocol, and so the Rank service is included as part of the HARD LINKS profile option.

A.4.2 **Expedited Data**

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

Expedited data is specified in Edition 3 Annex A in a manner that does not conflict with any of the Edition 4 specification. So use of the Edition 3 specification is straightforward. Key sections:

- Table A-1: summary of the S_EXPEDITED_UNIDATA service
- Sections A.2.1.10 – A.2.1.13: detailed definition of the expedited data services
- Sections A.3.1.1– A.3.1.3: how data is transferred, including normal and expedited data
- Section A.3.2.4: procedures for data transfer, including normal and expedited data

Annex B (CAS) does not distinguish between normal and expedited data. Expedited information is passed to Annex C,

A.5 Changes in Edition 4

The following changes are made relative to Edition 3.

1. The SIS protocol is moved into its own Annex (Annex S) rather than being split between Annex's A and F. This improves clarity of Annex A and allows it to follow OSI model of service specification with peer protocol.

   Hard Links are deprecated. These added significant complexity/confusion and are of no current or anticipated use.
2. Expedited data is deprecated. This is of no current or anticipated operational use. Expedited data was used in some OSI protocols, but is not used in modern applications. Edition 3 Annex H also recommends against its use. They are retained to ensure interoperability with Edition 3 systems.

3. Rank is removed. This apparently useful capability is confusing, as its use (excluding hard links) is only for two functions that are not needed:

   a. Restriction of Management Messages to Rank 15. This is a useless and strange control, as rank is client selected.
   b. Choice of client to use if resources are constrained (from Annex H). This is not relevant to a modern implementation.

4. Addition of new error types.
5. Clarification of queueing model to support ALE use as defined in Annex B.
6. Prevent use of “in-order” with Non-ARQ.

ANNEX B CHANNEL ACCESS SUBLAYER (mandatory)

The channel access sublayer handles links between peers.

B.1. Channel Access Sublayer Service Definition

The Channel Access Sublayer provides services to the Subnetwork Interface Sublayer. These services are:

1. Execute requests by the Subnetwork Interface sublayer to "*Make"* and *"Break"* Physical Links

2. Notify the Subnetwork Interface sublayer of changes in the state of a Physical Link.

3. Accept S_PDUs (encapsulated in the appropriate primitive) from the Subnetwork Interface sublayer for transmission on a Physical Link.

4. Deliver S_PDUs (encapsulated in the appropriate primitive) received on a Physical Link to the Subnetwork Interface sublayer.

The term "physical link" is used in STANAG 5066 to mean a logical link between a pair of nodes with consistent ARQ state.

In order to provide these services, the Channel Access Sublayer implements a protocol that specifies the tasks that must be executed and the rules that must be obeyed by the sublayer. While a number of different channel-access protocols are possible, the one that is suitable for this document is referred to as the Channel Access Type 1 (CAS-1) protocol, and described herein.

The Channel Access Sublayer also manages the operation of ALE links with peer nodes.

B.1.1. Changes in This Edition

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

The following key changes have been made:

1. The CAS-1 protocol has been extended to enable negotiation of support of this version of the protocol. For peers not supporting this version of the protocol, only Edition 3 compatible features are used.
2. ALE support has been added. In order to achieve this, various models of peer access are defined, which has added some new early sections to this document and various protocol changes.
3. Hard Links. These are now deprecated and specified in Section B.9.1, rather

than in the core specification.

B.2. Interface Primitives Exchanged with the Subnetwork Interface Sublayer

The implementation of the interface between the Channel Access Sublayer and the Subnetwork Interface 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 Subnetwork Interface 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 Channel Access Sublayer, i.e.:

1. the interface **shall** enable the Subnetwork Interface sublayer to submit requests to change the state of a physical link, i.e., to make or break a physical link with a specified node address;

2. The interface **shall** enable the Channel Access sublayer to notify the Subnetwork Interface sublayer of changes in the status of the physical link;

3. The interface **shall** allow the Channel Access sublayer to accept S_PDUs from the Subnetwork Interface sublayer;

4. The interface **shall** allow the Channel Access sublayer to deliver S_PDUs to the Subnetwork Interface sublayer.

Additionally, the protocol-control information from the Subnetwork Interface sublayer that is required for the management of the Channel Access sublayer **shall** not be derived from knowledge of the contents or format of any client data or U_PDUs encapsulated within the S_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 Channel Access Sublayer does not queue traffic. The DTS will queue traffic and in some modes of operation SIS will queue traffic. CAS can process

information with its peers and between the layers immediately, without need for any queueing.

The interface may use knowledge of the contents of S_PDUs (excluding the contents of any encapsulated U_PDUs) to derive protocol control information for the Channel Access sublayer. This approach is highly discouraged, however. The recommended approach for implementation is that information required for protocol control within the Channel Access 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 Channel Access Sublayer and the Subnetwork Interface 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, the technical specification of the Channel Access Sublayer assumes communication with the Subnetwork Interface sublayer using primitives prefixed with a “C_”. A minimal set of C_Primitives has been assumed that meet the requirements stated above and the general function for each C_Primitive is given in Table B-1. These C_Primitives are given without benefit of a list of arguments or detailed description of their use. As noted initially, they are offered for information only as a means of describing the interaction between the Channel Access sublayer and the Subnetwork Interface sublayer, and as general guidance for a possible implementation.

Table B-1- Nominal Definition of C_Primitives for the Interface between the Channel Access sublayer and the Subnetwork Interface sublayer (non-mandatory, for information-only)

NAME OF PRIMITIVE	DIRECTION (see Note)	COMMENTS
C_PHYSICAL_LINK_MAKE	SIS →CAS	request to make a physical link to a specified node
C_PHYSICAL_LINK_MADE	CAS→SIS	report that a physical link was made
C_PHYSICAL_LINK_REJECTED	CAS→SIS	report that a physical link could not be made and that the initiating request is
C_PHYSICAL_LINK_BREAK	SIS→CAS	request to break a physical link
C_PHYSICAL_LINK_BROKEN	CAS→SIS	report that a physical has been broken (by request, or unilaterally)
C_UNIDATA_REQUEST	SIS→CAS	delivers an S_PDU to the Channel Access Sublayer, requesting normal data-delivery
C_UNIDATA_REQUEST_CONFIRM	CAS→SIS	confirms S_PDU delivery to the remote node using the normal data-delivery
C_UNIDATA_REQUEST_REJECTED	CAS→SIS	notifies that an S_PDU could not be delivered to a remote node using the
C_UNIDATA_INDICATION	CAS→SIS	delivers an S_PDU that had been received using the normal delivery service to the Subnetwork Interface sublayer

[Note: SIS = Subnetwork Interface Sublayer; CAS = Channel Access Sublayer;
→ = direction of message flow from source to destination sublayer]

B.2.1. **Parameters to C_PHYSICAL_LINK_BREAK**

The C_PHYSICAL_LINK_BREAK primitive has a reason provided by SIS. This reason has the value “Soft Link Terminated for higher priority link” or “Soft Link Terminated to share channel” or “Idle Soft Link Terminated”. This reason is encoded as the Reason value in the PDU sent to the peer to break the link as specified in Section B.7.1.5.

B.2.2. **Errors from C_ Primitives**

The parameters of the C_ service primitives that are clear from the service definition are not listed explicitly. This section lists errors from selected service primitives.

B.2.2.1. **Errors from C_PHYSICAL_LINK REJECTED**

The errors are grouped by where they came from:

1. Rejection by peer with sub-reasons from PHYSICAL LINK REJECTED C_PDU:
    a. Reason Unknown
    b. Broadcast only node
2. DTS rejected transfer of Link Make using D_ EXPEDITED_UNIDATA_REQUEST_REJECTED. The DTS has no peer

mechanism for rejection. The only local mechanism for rejection is TTL expiry. TTL **should** be set to value large enough to ensure this does not happen in practice.

3. Timer expired after final transmission of PHYSICAL LINK REQUEST C_PDU.
4. Local Reasons, defined by this specification:
    a. “Maximum Physical Links In Use”.
    b. “ALE Call Timeout”
    c. “ALE Call Rejected”
    d. “Duplex ALE Timeout”
    e. “Link not allowed due to EMCON”
    f. “Physical Link Prevents Broadcast”

These errors are handled as follows:

1. Broadcast only reflects a mis-configuration which needs to be addressed.
2. “Maximum Physical Links In Use” will lead to SIS layer holding off on establishing a physical link to this peer until another physical link has been closed.
3. Other errors will lead to the SIS layer rejecting all S_PDUs queued for the link, and allowing the SIS user to resubmit the messages if it wishes.

B.2.2.2. **Errors from C_UNIDATA_REQUEST_REJECTED**

C_UNIDATA_REQUEST is an unacknowledged service, and so there are no peer errors. The following errors are noted:

1. DTS rejected transfer D_ UNIDATA_REQUEST_REJECTED. The DTS has no peer mechanism for rejection. The only local mechanism for rejection of a valid PDU is:
    a. TTL expiry.
2. Local reasons, defined in this specification
    a. Broadcast Not Allowed
    b. Address Not Known
    c. No ALE Mapping for Address
    d. Physical Link Needed
    e. Address not Routable

TTL Expiry and the first three local reasons will be passed up to the SIS user along with rejection of the S_PDU. The “Physical Link Needed” error is used when Non-

ARQ data is sent to a peer using ALE, where it is necessary to establish a physical link for all traffic to the peer.

B.3. Interface Primitives Exchanged with the ALE Layer

When ALE is used there is a direct service interaction between the ALE layer and CAS. The primary service elements used are defined in Table B-2. These primitives are used to specify the CAS/ALE interaction. There is no requirement for an implementation to follow this service model.

Table B-2- Nominal Definition of ALE_Primitives for the Interface between the Channel Access sublayer and ALE (non-mandatory, for information-only)

NAME OF PRIMITIVE	DIRECTION	COMMENTS
ALE_LINK_MAKE	CAS →ALE	request to make an ALE link to the node or list of nodes specified
ALE_LINK_COMPLETED	ALE→CAS	report that an ALE link has been made, in response to a local ALE_LINK_MAKE request
ALE_LINK_REJECTED	ALE→CAS	report that a requested ALE link could not be made, typically because of timeout or not available HF frequency
ALE_LINK_REMOTE	ALE→CAS	report that an ALE link has been made, when another node has initiated a link
ALE_LINK_TERMINATE	CAS→ALE	request to terminate the active ALE link
ALE_LINK_TERMINATED	ALE→CAS	report that the active ALE link has been terminated, either by a local operator or by another node in the link
ALE_LINK_RELINK	CAS->ALE	some ALE subsystems will support an option to re-link using ALE Link Management (ALM)

[Note: CAS = Channel Access Sublayer; ALE = ALE Layer
→ = direction of message flow from source to destination sublayer]

This model assumes that ALE searching is controlled by the ALE layer; it is anticipated that the ALE layer will usually search when there is no active link or no link has been requested.

This model does not include sounding, which is discussed in Section B.6.2.7.

B.4. Models of Peer Access

There are two models of channel access, described in the next two sections:

1. Multiple Simultaneous Peer Access

2. Single Peer Access

The model in use **shall** be the same for all nodes using a channel.

For a profile of this standard, the model or models supported need to be specified.

B.5. Multiple Simultaneous Peer Access

The first model is that a node on a channel can simultaneously transmit to multiple peers, utilizing the broadcast nature of the underlying channel. Control of access to the channel is then managed by the MAC layer using CSMA (Annex K) or Wireless Token Ring Protocol (Annex L).

In this model, the Channel Access Sublayer can submit data for all destinations directly to the DTS.

Support of this model is optional. Support of this option by an implementation is indicated with the MULTIPLE ACCESS profile option.

B.6. Single Peer Access

In this second model, the Channel Access Sublayer accesses a channel or channels that can communicate with just one peer. This model **shall** be used in two situations.

1. Where Automatic Link Establishment (ALE) is used to link between peers. When used, this ALE mode of operation **shall** be configured for all nodes accessing the channel; and/or
2. Duplex operation, where exchange of data between two nodes use two exclusive channels, one for transmission and one for reception.

In this model, the Channel Access Sublayer will control the active peer (or active peers with each peer on an independent channel). Requests for transfers to other peers will be rejected, requiring the SIS to queue data and manage traffic for multiple peers where the number of channels available is less than the number of peers for which there is traffic.

This model has four detailed variants set out below. Each of these variants is optional and support is indicated with a profile option.

B.6.1. Duplex without ALE

When Duplex configuration is used without ALE, a pair of channels need to be set up between two nodes (one for each direction of transfer). This setup is only viable for one pair of nodes.

Support of this model is optional. Support of this option by an implementation is indicated with the DUPLEX FIXED profile option.

For this configuration, Channel Access operates following the procedures of multiple simultaneous peer access. No queuing in the SIS layer is needed, as there is only one peer. One additional check is made in this configuration:

1. If data is addressed to a unicast STANAG 5066 address other than that of the single duplex peer, it **shall** be rejected with reason code “Address Not Known”

Group addressing **may** be used. Although it is not usually of operational benefit when there is a single peer, it may be helpful for some applications.

B.6.2. Operation with 1:1 ALE

A channel may be configured to use ALE (Automatic Link Establishment) to connect to one peer at a time. The procedure defined here can be used with any ALE protocol, including 2G, 3G and 4G. When ALE is used, it **shall** be used for all communication.

Support of this model is optional. Support of this option by an implementation is indicated with the ALE 1:1 profile option.

B.6.2.1. ALE 1:1 Modes

1:1 ALE has a choice of two modes:

1. IMPICIT CAS-1. In this mode, a CAS-1 soft link is implicitly established when the ALE link is made and implicitly closed with the ALE link ends. This mode optimizes performance where ALE/CAS-1 links are short and there is usually not a need to relink ALE.
2. EXPLICIT CAS-1. In this mode, a CAS-1 soft link is explicitly negotiated by protocol exchange after the ALE link is established. If the 1:1 ALE mode is supported, this option **shall** be supported. This mode optimizes performance where CAS-1 links are long and commonly relink ALE during the CAS-1 link. It enables a CAS-1 link with a peer to be maintained over an extended period.

The choice of mode is determined by the STANAG 5066 Profile, as defined in the core specification. The mode **shall** be the same for all nodes in a network.

The choice of mode may be influenced by interoperability with Edition 3 systems, as described in Section B.7.2.3.

B.6.2.2. Queueing Model

When IMPLICIT CAS-1 is used, there is a single active CAS-1 link. Queueing for this single link and switching between queues is modelled in SIS and specified in Annex A, so that the SIS service controls which link is active.

For EXPLICIT CAS-1, there may be multiple CAS-1 links open and the active link is not visible at the SIS layer. For EXPLICIT CAS-1, the CAS needs to manage an independent queue and DTS state for each active CAS-1 link. The CAS may open and close ALE links as it chooses, associating the correct queue/DTS state with the open ALE link.

B.6.2.3. Handling Multiple ALE 1:1 Channels

A node **may** be configured to have multiple channels of this nature, although it is expected that most nodes will be single channel. Where there are multiple channels, the DTS for each channel **shall** operate independently. Each channel in a multi channel system will need a separate modem and ALE unit with an independent DTS. This setup can be considered as a single CAS layer managing multiple modem/ALE/DTS stacks.

B.6.2.4. Procedure of Operation

When data arrives for a peer (ARQ or non-ARQ), the following checks are made:

1. If the data is addressed to a broadcast address, it **shall** be rejected with reason code “Physical Link Prevents Broadcast” if a 1:1 ALE link is open. If it is not open, it **may** be handled by the procedure of Multicast ALE set out in Section B.6.3.
2. If data is addressed to an unknown STANAG 5066 address, it **shall** be rejected with reason code “Address Not Known”
3. If an ALE address cannot be determined for the address, it **shall** be rejected with reason code “No ALE Mapping for Address”

When an ALE connection is established to a peer, a CAS-1 link is always established for the peer. In EXPLICIT CAS-1 mode, the CAS-1 soft link establishment procedure described below **shall** be followed, noting that if a CAS-1 link is already open that an ALE link can be established directly. In IMPLICIT CAS-1 mode, the CAS-1 soft link establishment procedure described below **may** be followed, either to determine which edition of STANAG 5066 a peer supports or to negotiate the optional use of short frame sequence number D_PDUs between Edition 4 peers. As an ALE connection initializes the CAS-1 link, the CAS-1 initialization procedure does not need to be used. It is recommended that the CAS-1 link process is only used if necessary.

When closing an ALE link with a peer, the ALE Link Break procedure specified in this Annex **shall** be followed. In EXPLICIT CAS-1 mode, the ALE link **may** be broken with the CAS-1 link remaining open. To close CAS-1 link in EXPLICIT

mode, the CAS-1 link break procedure **shall** be used. In IMPLICIT CAS-1 mode, the CAS-1 link break procedure **should not** be used; Break of the CAS-1 link is implicit from ALE Link Break.

If the ALE link is closed externally (e.g., by an operator) the ALE Link Break procedure **shall** be followed, provided that the ALE link is open. In IMPLICIT mode, the CAS-1 link **shall** be treated as closed when the ALE link is closed..

The Channel Access Sublayer **shall** maintain a list of open physical links with maximum number of physical links the same as the number of channels. Each channel will have an independent DTS. When all channels have assigned physical links, any request to open a new physical link **shall** be rejected with error “Maximum Physical Links In Use”. Commonly this will be limited to one channel and one active physical link.

When ALE is in use, non-ARQ traffic will need to use an appropriate ALE link. To achieve this in IMPLICIT mode, a physical link is mandated for non-ARQ traffic as well as ARQ traffic. In the event of non-ARQ traffic being submitted when there is not a physical link to the destination, the submission **shall** be rejected with reason “Physical Link Needed”.

In EXPLICIT mode, the CAS layer **shall** ensure that an appropriate ALE link is open to carry the non-ARQ traffic.

In normal operation of 1:1 ALE in IMPLICIT mode, the SIS layer will control opening and closing of CAS-1 links to fit with traffic load. When a CAS-1 link is opened, this will lead to an ALE link being established. The CAS-1 Link will be closed cleanly based on Clean Close State specified in Section B.7.2.4. ALE mechanisms are used to open and close the link and CAS-1 link open and close is implicit. This is desirable, as it removes the unnecessary overhead of CAS-1 protocol handshaking.

There are two situations where this basic model of one ALE link per CAS-1 link needs to be extended, which are described in the following two sections.

B.6.2.5. **Frequency Change/Validation**

For a long running ALE link, a STANAG 5066 application may determine (automatically or with operator input) that operation on a different frequency may be beneficial or choose to validate the current frequency. The approach taken depends on the mode.

For EXPLICIT CAS-1, the current ALE link is either re-linked using Automatic Link Management (ALM) or the link is closed and re-opened.

For IMPICIT CAS-1, the re-linking procedure described in Section B.7.2.7is followed.

B.6.2.6. ALE Link Failure

A second scenario requiring a new ALE link is when an ALE link fails. This will typically be caused by a fading channel that leads to communication being lost between the two end point, both of which will time out the ALE link. The approach taken depends on the mode.

For EXPLICIT CAS-1, a new ALE link is established and the CAS-1 soft link continue.

For IMPICIT CAS-1, the procedure for resuming a link described in Section B.7.2.9 is followed.

B.6.2.7. ALE Sounding

In order to operate efficiently, ALE needs to sound on the frequencies used in order to determine which frequency is best. In a STANAG 5066 & ALE system with light load, it will often be possible and sensible for the ALE component to manage sounding independent of STANAG 5066.

In a highly loaded system, the ALE module may not get sufficient access to the modem to independently perform such sounding. In such a setup, it will often be important for the STANAG 5066 system to pro-actively manage and co-ordinate ALE sounding. The details of achieving this are outside the scope of this specification.

B.6.3. Operation with Multicast ALE (non-ARQ)

It is also possible to use ALE to open a link to multiple nodes. This **may** be done when multicast traffic is processed, for example in support of a multicast protocol such as ACP 142. This link is used for non-ARQ traffic only.

NOTE: While it is technically possible to manage soft links over multicast ALE, the process for managing this appears unduly complex, so this is not supported in this edition of STANAG 5066.

Support of this model is optional. Support of this option by an implementation is indicated with the MULTICAST ALE profile option.

Multicast ALE is initiated when non-ARQ data is sent to a STANAG 5066 broadcast address. The ALE call addresses used will be determined from the STANAG 5066 broadcast address and an appropriate ALE Group Call initiated.

In order to use Multicast ALE, any 1:1 ALE links need to be closed first. The SIS layer has knowledge of this requirement and will close down any soft links and associated ALE 1:1 links before using the Multicast ALE service.

Multinode ALE calls need to be closed by the call initiator. This may be done after an idle period or when the SIS layer requests establishment of a soft link and associated 1:1 ALE link.

B.6.4. **Duplex with 1:1 ALE**

In order to use Duplex transfer between a pair of nodes without ALE, it is necessary to configure a fixed frequency for each direction of transfer between a pair of nodes.

Support of this model is optional. Support of this option by an implementation is indicated with the DUPLEX WITH ALE profile option.

When a pair of nodes each have duplex support and ALE is used, duplex communication can be negotiated. In order to do this, a node shall determine if the peer can support duplex communication. There are two methods to achieve this:

1. A priori knowledge; or
2. The Data Transfer Sublayer enables determination that a peer node is able to support duplex by use of EOWs.

If a node determines that both peers support duplex, duplex communication **may** be negotiated using ALE. It is recommended that this is done. On receiving an ALE link, a node with a duplex peer **may** use ALE to negotiate a second connection with the peer. If this is successfully done, the first ALE link **shall** be used by the node which initiated it to send data and the second ALE link **shall** be used to send data in the reverse direction.

B.7. Channel Access Protocol Type 1 and C_PDUs

Where ALE is not used, the local node will be connected to a channel, which can be fixed frequency or with variable frequency determined by an external process (e.g., selection of frequency based on a fixed schedule). Other nodes may be connected to the same channel.

The local node **may** know the addresses of all other nodes that can be connected to the channel. If this is the case and data is addressed to another node, it **shall** be rejected with reason code “Address Not Known”.

The co-ordination of the making and breaking of Physical Links (hereinafter referred to as 'CAS 1 Linking Protocol') between two nodes **shall** be performed solely by the Channel Access Sublayer. If ALE is not used, the CAS 1 linking protocol **shall** be used.

If the CAS 1 Linking Protocol is omitted (when ALE is used) and the response to ARQ-data DPDUs is a Warning DPDU indicating that a connection is not made, then the CAS 1 Linking Protocol **shall** be followed and the data resent.

If a slave node (receiving station) accepts an ALE link and a CAS-1 Physical-Link Request C_PDU is received, then the receiving node **shall** respond in accordance with the requirements of the 'CAS 1 Linking Protocol', accepting or rejecting the request as is appropriate for its state.

When ALE is used, the CAS 1 linking protocol **shall** respond to the reception of a type 4 PHYSICAL_LINK_BREAK C_PDU by sending a type 5 PHYSICAL_LINK_BREAK_CONFIRM C_PDU as specified in Section B.4.2.2. The ALE link is then closed and the physical link **shall** be broken at that point.

A Node shall use a Physical Link to support control and data exchange for Soft-Link Data Exchange Sessions as requested by the Subnetwork Interface Sublayer.

A Node **may** have Physical Links with more than one other node at a time, up to one Physical Link per remote node; i.e., a Node may "Make" a new Physical Link with another node before it "Breaks" any Physical links.

There **shall** be no explicit peer-to-peer communication required to switch from use of one Physical Link to another.

When 1:1 ALE is used, the Channel Access sublayer enforces that there is at most one active ALE link per channel and that for each active ALE link there is a single physical link associated with that ALE link.

The Channel Access Sublayer CAS 1 linking protocol **shall** communicate with peer sublayers in other nodes using the protocols defined here in order to:

1. Make and break physical links

2. Deliver S_PDUs between Subnetwork Interface Sublayers at the local node and remote node(s).

B.7.1. **Type 1 Channel Access Sublayer Data Protocol Units (C_PDUs)**

The following C_PDUs **shall** be used for peer-to-peer communication between Channel Access Sublayers in the local and remote node(s):

C_PDU NAME	*Type*
DATA C_PDU	TYPE 0
PHYSICAL LINK REQUEST	TYPE 1
PHYSICAL LINK ACCEPTED	TYPE 2
PHYSICAL LINK REJECTED	TYPE 3
PHYSICAL LINK BREAK	TYPE 4
PHYSICAL LINK BREAK CONFIRM	TYPE 5
PHYSICAL LINK BREAK REQUEST	TYPE 6
PHYSICAL LINK RESUME	TYPE 7
PHYSICAL LINK RESUME	TYPE 8
PHYSICAL LINK RESUME REJECT	TYPE 9
ALE RELINK	TYPE

The first argument and encoded field of all C_PDUs **shall** be the C_PDU Type

The remaining format and content of these C_PDUs **shall** be as specified in the subsections that follow.

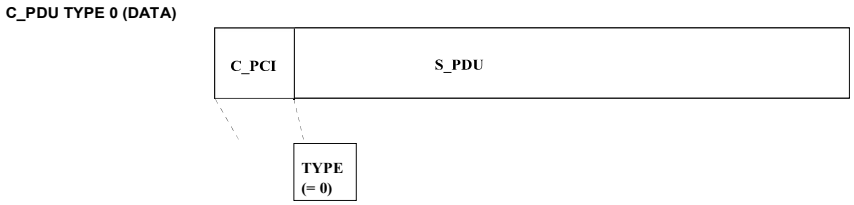
Unless noted otherwise, argument values encoded in the C_PDU bit-fields **shall** be mapped into the fields in accordance with CCITT V.42, 8.1.2.3, i.e.:

1. when a field is contained within a single octet, the lowest bit number of the field **shall** represent the lowest-order (i.e., least-significant-bit) value;
2. when a field spans more than one octet, the order of bit values within each octet **shall** decrease progressively as the octet number increases. The lowest bit number associated with the field **shall** represent the lowest-order (i.e., least significant bit) value.

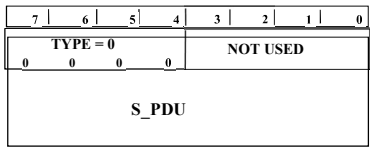
Unless noted otherwise, bit-fields specified as NOT USED **shall** be encoded with the value '0' (i.e., zero).

B.7.1.1. **DATA C_PDU**

Type :
"0" = DATA C_PDU
Encoding :



(a) Generic Encoding



(b) Bit-Field Map

Figure B-1: Generic Encoding and Bit-Field Map of the DATA C_PDU

Description :

The DATA (TYPE 0) C_PDU **shall** be used to send an encapsulated S_PDU from the local node to a remote node.

The *Type* argument **shall** be encoded in the first four-bit field of the DATA C_PDU as shown in Figure B-1. The value of the *Type* argument for the DATA C_PDU **shall** be zero.

The remaining octets of the DATA C_PDU **shall** contain the encapsulated S_PDU and only the encapsulated S_PDU.

For the Channel Access sublayer request to the lower layers of the subnetwork to deliver a C_PDU, the delivery service requirements for a DATA C_PDU **shall** be the same as the S_PDU that it contains, i.e.:

a) C_PDUs **shall** be sent using the normal Data Delivery service provided by the lower sublayer;
b) the DELIVERY mode specified by the Subnetwork Interface Sublayer for the encapsulated S_PDU (i.e., ARQ, non-ARQ, etc.) also **shall** be assigned to the C_PDU by the Channel Access sublayer as the delivery mode to be provided by the lower sublayer.

B.7.1.2. PHYSICAL LINK REQUEST C_PDU

Type :

"1" = PHYSICAL LINK REQUEST

Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=1				Not Used	Short	Ed4	LINK

Figure B-2: Generic Encoding and Bit-Field Map of the PHYSICAL LINK REQUEST C_PDU

Description :

The PHYSICAL LINK REQUEST C_PDU **shall** be transmitted by a Channel Access sublayer to request the making of the Physical Link.

The PHYSICAL LINK REQUEST C_PDU **shall** consist of the arguments *Type* and *Link*.

The value of the *Type* argument for the PHYSICAL LINK REQUEST C_PDU **shall** be '1' (i.e., one), encoded as a four-bit field as shown in Figure B-2.

The bits not used in the encoding of the PHYSICAL LINK REQUEST C_PDU **shall** be reserved for future use and not used by any implementation.

The Ed4 bit indicates to the peer that the local node support edition 4 (or subsequent version) of the protocol. If the peer is known to support edition 4 (or subsequent) this bit **shall** be set. If the peer is known to not support edition 4 (or subsequent) this bit **shall** not be set. Otherwise this bit **may** be set in order to discover if the peer supports edition 4.

If the Ed4 bit is set, the Short bit **may** be set. This indicates a request to use D_PDUs with short LFSN, which can give performance advantage at very low speed.

The LINK bit was used in Edition 3 to request an exclusive link. This capability is not supported by default in Edition 4. If the HARD LINKS deprecated optional profile is supported, then it **shall** be used as specified in Section B.9.1. Otherwise, this bit shall be set to zero on transmission. Error Code 5 (Exclusive Links Not Supported) shall be used in response to a request with this bit set.

When a PHYSICAL LINK REQUEST C_PDU is transmitted, the local Node is not linked to another Node and therefore the ARQ transmission mode is not supportable by the Data Transfer Sublayer.
Therefore, the PHYSICAL LINK REQUEST C_PDU **shall** be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

A Channel Access sublayer which receives a PHYSICAL LINK REQUEST C_PDU **shall** respond with either a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU or a PHYSICAL LINK REJECTED (TYPE 3) C_PDU, as appropriate.

B.7.1.3. PHYSICAL LINK ACCEPTED C_PDU

Type :

"2" = PHYSICAL LINK ACCEPTED

Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=2				Not Used	Short	Ed4	LINK

Figure B-3: Generic Encoding and Bit-Field Map of the PHYSICAL LINK ACCEPTED C_PDU

Description :

The PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU **shall** be transmitted by a peer sublayer as a positive response to the reception of a TYPE 1 C_PDU (PHYSICAL LINK REQUEST).

PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU **shall** consist only of the argument *Type*. [Note: A Node can never request the establishment of more than one Physical Link with another given node, therefore, the request for which this message is a response is uniquely identified by the node address of the node to which the PHYSICAL LINK ACCEPTED C_PDU is sent.]

The *Type* argument **shall** be encoded as a four-bit field containing the binary value 'two' as shown in Figure B-3.

The bit not used in the encoding of the PHYSICAL LINK ACCEPTED C_PDU **shall** be reserved for future use and not used by any implementation. They **shall** be set to zero on transmission.

The LINK bit was used in Edition 3 to request an exclusive link. This capability is not supported by default in Edition 4. If the HARD LINKS deprecated optional profile is supported, then it **shall** be used as specified in Section B.9.1. Otherwise, this bit shall be set to zero on transmission.

The Ed4 bit indicates to the peer that the local node support edition 4 (or subsequent version) of the protocol. If the equivalent bit was set in the PHYSICAL LINK REQUEST C_PDU which is being responded to this bit **shall** be set. Otherwise, this bit **shall not** be set.

If the Ed4 bit is set and the Short bit was set in the PHYSICAL LINK REQUEST C_PDU which is being responded to, the Short bit **may** be set. This indicates that

D_PDUs with short LFSN, which can give performance advantage at very low speed **shall** be used with this link in both directions. If Ed4 bit is set and the Short bit is not set, D_PDUs with short LFSN **shall not** be used. This means that between Ed4 peers, short LFSN is only used if both peers agree.

When a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supportable by the Data Transfer Sublayer. Therefore, the PHYSICAL LINK ACCEPTED C_PDU **shall** be sent by the Channel Access Sublayer requesting the lower-layer's Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

B.7.1.4. PHYSICAL LINK REJECTED C_PDU

Type :

"3" = PHYSICAL LINK REJECTED

Encoding :

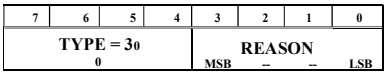


Figure B-4: Generic Encoding and Bit-Field Map of the PHYSICAL LINK REJECTED C_PDU

Description :

The PHYSICAL LINK REJECTED (TYPE 3) C_PDU **shall** be transmitted by a peer sublayer as a negative response to the reception of a TYPE 1 C_PDU (PHYSICAL LINK REQUEST).

The PHYSICAL LINK REJECTED (TYPE 3) C_PDU **shall** consist of two arguments: *Type* and
Reason.

The *Type* argument **shall** be encoded as a four-bit field containing the binary value 'three' as shown in Figure B-4.

The *Reason* argument **shall** be encoded in accordance with Figure B-4 and the following table:

Reason	Value
Reason Unknown	0
Broadcast-Only-Node	1
Higher-Priority-Link-Request-Pending	2

Not Used	3
Too many active links	4
Exclusive Links Not	5
unspecified	6-15

The value 0, indicating an unknown reason for rejecting the Make request for the physical link, is always a valid reason for rejection. Reasons corresponding to values in the range 6-15 are currently unspecified and unused in the STANAG.

When a PHYSICAL LINK REJECTED C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supportable by the Data Transfer

Note that “Link busy” is not included in the list of possible reasons for rejecting anode’s request to establish a link since this function is provided by the WARNING frame type of the Data Transfer Sublayer, as specified in Annex C.

The PHYSICAL LINK REJECTED C_PDU **shall** be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

B.7.1.5. PHYSICAL LINK BREAK C_PDU

Type :

“4” = PHYSICAL LINK BREAK

Encoding :

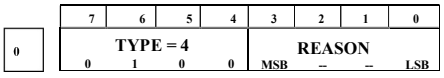


Figure B-5: Generic Encoding and Bit-Field Map of the PHYSICAL LINK BREAK C_PDU

Description :

The PHYSICAL LINK BREAK C_PDU **shall** be transmitted by either of the peer Channel Access sublayers involved in an active Physical Link to request the breaking of the link.

The PHYSICAL LINK BREAK C_PDU **shall** consists of two arguments: *Type and Reason.*

The *Type* argument **shall** be encoded as a four-bit field containing the binary value 'four' as shown in Figure B-5.

The *Reason* argument **shall** be encoded in accordance with Figure B-5 and the following table:

Reason	Value
Reason Unknown	0
Higher-Layer-	1
Switching to Broadcast- Data-	2
Switching link to send higher priority data	3
No More Data	4
Switching link for	5
Operator request	6
unspecified	7-15

The value 0, indicating an unknown reason for requesting the breaking of a physical link, is always a valid reason. Reasons corresponding to values in the range 5-15 currently are not defined in this STANAG.

A peer sublayer which receives the PHYSICAL LINK BREAK C_PDU **shall** immediately declare the Physical Link as broken and respond with a PHYSICAL LINK BREAK (TYPE 5) C_PDU as specified in section B.3.1.6 below.

The PHYSICAL LINK BREAK C_PDU **shall** be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.
[Note: The reason the PHYSICAL LINK BREAK C_PDU is sent with the non-ARQ data service, even though a physical link exists at the time that it is sent, is because the receiving peer will immediately declare the Link as broken. The receiving peer will therefore not have time to send ARQ acknowledgements for the C_PDU.]

B.7.1.6. PHYSICAL LINK BREAK CONFIRM C_PDU

Type :

"5" = PHYSICAL LINK BREAK CONFIRM

Encoding :

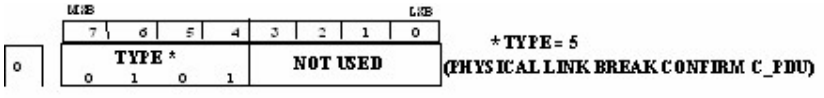


Figure B-6: Generic Encoding and Bit-Field Map of the PHYSICAL LINK BREAK CONFIRM C_PDU

Description :

The PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU **shall** be transmitted by a Channel Access sublayer as a response to a TYPE 4 "PHYSICAL LINK BREAK" C_PDU.

The PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU **shall** consist of the single argument *Type*.

The *Type* argument **shall** be encoded as a four-bit field containing the binary value 'five' as shown in Figure B-6.

The PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU **shall** be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

Upon receiving a PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU, the peer which initiated the breaking of the Link **shall** declare the Link as broken.

B.7.1.7. PHYSICAL LINK BREAK REQUEST C_PDU

Type :

"6" = PHYSICAL LINK BREAK REQUEST

Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=6				Not Used	Reason		
1	Priority							

Figure B-7: Encoding of the PHYSICAL LINK BREAK REQUEST C_PDU

Description :

The PHYSICAL LINK BREAK REQUEST (TYPE 6) C_PDU **may** be transmitted by a peer sublayer to indicate to the peer a desire to break the link. This enables a link to be closed cleanly, with co-operation between both ends resulting in the clean close state specified in Section B.7.2.4.

This C_PDU is defined in Edition 4 and **shall not** be transmitted to Edition 3 peers.

This C_PDU is used to request termination of ALE link and associated physical link.

The three bit Reason field indicates the reason that the break is being requested. MSB of Reason is bit 2 and LSB is bit 0. Values for Reason are specified in the following table:

Reason	Value	Description
No Data	000	The node sending PHYSICAL LINK BREAK REQUEST has no more data to send and suggests the link is terminated
Higher Priority	001	There is data for another destination of higher priority than traffic for the current link and link needs to be closed to transmit it.
Channel Share	010	Link needs to be closed so that the channel can be shared fairly with another link.
Operator Request	011	An operator has requested that the link be broken
Not used	100 - 111	Reserved for future versions of this protocol.

The Priority field **shall** be set to the priority of the higher priority traffic when Reason is

"Higher Priority". For other Reasons, the priority field **shall** be set to 0.

When a PHYSICAL LINK BREAK REQUEST (TYPE 6) C_PDU is transmitted the local Node is linked to the peer Node. Therefore, the PHYSICAL LINK BREAK REQUEST C_PDU **shall** be sent by the Channel Access Sublayer requesting the lower-layer's ARQ Data Services in accordance with the Annex C specification of the Data Transfer Sublayer. This will mean that the PHYSICAL LINK BREAK REQUEST C_PDU will arrive after any data sent in the same transmission. Before the peer breaks the link, it will need to ensure that all data prior to the PHYSICAL LINK BREAK REQUEST C_PDU has been received, which can be done by examination of the DTS ARQ window.

B.7.1.8. PHYSICAL LINK RESUME C_PDU

Type :
"7" = PHYSICAL LINK RESUME
Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=7				Not Used			

Figure B-8: Encoding of the PHYSICAL LINK RESUME C_PDU

Description :
The PHYSICAL LINK RESUME (TYPE 7) C_PDU **shall** be transmitted by a peer sublayer to indicate to the peer that it is desired to resume a physical link.

This C_PDU is defined in Edition 4 and **shall not** be transmitted to Edition 3 peers.

B.7.1.9. PHYSICAL LINK RESUME ACCEPT C_PDU

Type :
"8" = PHYSICAL LINK RESUME ACCEPT
Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=8				Not Used			

Figure B-9: Encoding of the PHYSICAL LINK RESUME ACCEPT C_PDU

Description :
The PHYSICAL LINK RESUME ACCEPT (TYPE 8) C_PDU **shall** be transmitted

by a peer sublayer to indicate to the peer that it is agreed to resume a physical link.

This C_PDU is defined in Edition 4 and **shall not** be transmitted to Edition 3 peers.

B.7.1.10. PHYSICAL LINK RESUME REJECT C_PDU

Type :
"9" = PHYSICAL LINK RESUME REJECT
Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=9				Not Used			

Figure B-10: Encoding of the PHYSICAL LINK RESUME REJECT C_PDU

Description :
The PHYSICAL LINK RESUME REJECT (TYPE 9) C_PDU **shall** be transmitted by a peer sublayer to indicate to the peer that it will not resume a physical link.

This C_PDU is defined in Edition 4 and **shall not** be transmitted to Edition 3 peers.

B.7.1.11. ALE RELINK C_PDU

Type :
"10" = ALE RELINK
Encoding :

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type=10				Not Used	Reason		

Figure B-11: Encoding of the ALE RELINK C_PDU

Description :
The ALE RELINK (TYPE 10) C_PDU **may** be transmitted by a peer sublayer to indicate to the peer a desire to break an ALE link, while retaining the physical link.

This C_PDU is defined in Edition 4 and **shall not** be transmitted to Edition 3 peers.

The three bit Reason field indicates the reason that the relink is being requested. MSB of Reason is bit 2 and LSB is bit 0. Values for Reason are specified in the following table:

Reason	Value	Description
ALE Quality	000	The current link is determined to be poorer quality than expected. Link needs to be closed so that ALE can see if a better channel is available.
ALE Time	001	ALE link has been running for a long time. Link needs to be closed so that ALE can see if a better channel is available.
Not used	010 - 111	Reserved for future versions of this protocol.

When a ALE RELINK (TYPE 10) C_PDU is transmitted the local Node is linked to the peer Node. Therefore, the ALE RELINK C_PDU **shall** be sent by the Channel Access Sublayer requesting one of the lower-layer's Expedited ARQ Data Services in accordance with the Annex C specification of the Data Transfer Sublayer. This well enable relinking to be performed as early as possible.

B.7.2. **Type 1 Channel Access Sublayer Peer-to-Peer Communication Protocol**

Requirements on the Type 1 Channel Access Sublayer Peer-to-Peer Communication Protocols are specified in this section and the subsections below.

The Channel Access sublayer **shall** perform all peer-to-peer communications for protocol control using the following C_PDUs:

C_PDUs for Peer-to-Peer Protocol	*Type*
PHYSICAL LINK REQUEST	TYPE 1
PHYSICAL LINK ACCEPTED	TYPE 2
PHYSICAL LINK REJECTED	TYPE 3
PHYSICAL LINK BREAK	TYPE 4
PHYSICAL LINK BREAK CONFIRM	TYPE 5

All peer-to-peer communications **shall** be done by the Channel Access sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs in accordance with the Annex C specification of the Data Transfer Sublayer).

The criteria for accepting or rejecting Physical Links Requests **shall** be as follows. A request to establish a Physical Link **shall** be accepted in Single Peer Access mode, as receipt of the request implies that the 1:1 underlying channel is open. In

Multiple Simultaneous Peer Access mode, the Physical Link **shall** be accepted, unless the number of active physical links exceeds a configurable limit or another operational reason.

B.7.2.1. Protocol for Making a Physical Link

In the specification that follows of the protocol for making a physical link, the node which requests the physical link is referred to as the Caller or Calling node, while the node which receives the request will be referred to as the Called node.

The protocol for making the physical link **shall** consist of the following steps:

Step 1-Caller:

Initiation of the link is in response to C_PHYSICAL_LINK_MAKE request from SIS,

If ALE is being used, the following initial caller procedure **shall** be used:

a) If ALE is being used, an ALE link to the peer **shall** be requested using ALE_LINK_MAKE.
b) If the ALE link is rejected or timed out with ALE_LINK_REJECT, the request for physical link **shall** be rejected with reason “ALE Call Timeout” or “ALE Link Rejected”. If ALE_LINK_COMPLETE is received, the link is made.
c) If Duplex ALE is being used, wait for the incoming ALE call indicated by ALE_LINK_REMOTE. If this does not arrive within a configurable time limit the first ALE call **shall** be released using ALE_LINK_TERMINATE and the request for physical link **shall** be rejected with reason “Duplex ALE Timeout”.
d) If the Ed4 support status of the peer is not known or if it is desired to use short frame sequence number D_PDUs to an Ed4 peer, the second stage of this procedure **shall** be followed. Otherwise in IMPLICIT CAS-1 mode, C_PHYSICAL_LINK_MADE is sent to the SIS, D_CONNECTION_MADE is sent to the DTS, and the link is made.

The following procedure is followed when ALE is not being used, or in EXPLICIT CAS-1 mode, or when the initial ALE procedure requires full CAS-1. This procedure is referenced above as “second stage”.

a) The Calling Node’s Channel Access Sublayer shall send a PHYSICAL LINK REQUEST (TYPE 1) C_PDU to initiate the protocol,
b) Upon sending the PHYSICAL LINK REQUEST (TYPE 1) C_PDU the Channel Access Sublayer shall start a timer which is set to a value greater than or equal to the maximum time required by the Called Node to send its response (this time depends on the modem parameters, number of re-transmissions, etc.),
c) The maximum time to wait for a response to a PHYSICAL LINK

REQUEST (TYPE 1) C_PDU shall be a configurable parameter in the implementation of the protocol.

Step 2-Called:

If ALE is being used, the called node waits for an incoming ALE link and receipt of ALE_LINK_COMPLETE from ALE to CAS. If duplex ALE is being used for the peer, the second ALE call is initiated using ALE_LINK_MAKE. If this second call fails, the initial ALE call is cleared using ALE LINK TERMINATE. On ALE completion D_CONNECTION_MADE is signaled to the DTS for each connected ALE peer.

Then an inbound C_PDU is waited for. C_PDUs will be received as a data indications from the DTS, noting that DATA C_PDUs will arrive as D_UNIDATA_INDICATION and other C_PDUs will arrive as D_EXPEDITED_UNIDATA_INDICATION.

- If it is a DATA C_PDU or a PHYSICAL LINK ACCEPTED C_PDU, the link is indicated made by:
  - SIS is informed that the CAS-1 soft link is open use C_PHYSICAL_LINK_MADE; and
  - The DATA C_PDU is passed to SIS.
  - NOTE: Receipt of PHYSICAL LINK ACCEPTED C_PDU is not expected. In the event that it is received, it is seen as preferable to accept the link rather than to clear it.
- If it is a PHYSICAL LINK REQUEST C_PDU, the procedure described below for handling PHYSICAL LINK REQUEST C_PDU is followed. If it is any other C_PDU, the ALE link is cleared.

If no C_PDU arrives within a configurable timer, the ALE link **shall** be cleared using ALE_LINK_TERMINATE.

The following procedure is followed when ALE is not used or when a PHYSICAL LINK REQUEST C_PDU arrives over an ALE link.

a) On receiving a PHYSICAL LINK REQUEST (TYPE 1) C_PDU, a Called node **shall**
determine whether or not it can accept or reject the request as follows:
   (1) If the Called node has the maximum number of active Physical Links, the Called node **shall** reject the request with reason “Too many active links”,
   (2) otherwise, the Called node **shall** accept the request for a Physical Link.

b) After determining if it can accept or reject the PHYSICAL LINK REQUEST, a Called node
shall respond as follows:
   (1) if a Called node accepts the physical link request, it **shall** respond with a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU,
   (2) otherwise, when a Called node rejects the physical link request, it **shall** respond with a PHYSICAL LINK REJECTED (TYPE 3)

C_PDU.

b) After a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU is sent, the called channel access sublayer **shall** declare the physical link made and transition to a state in which it executes the protocol for data exchange using C_PDUs.
c) If further PHYSICAL LINK REQUEST (TYPE 1) C_PDUs are received from the same address after the link is made, the Channel Access sublayer **shall** again reply with a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU.
d) If at least one DATA (TYPE 0) C_PDU is not received on a newly activated Physical Link after waiting for a specified maximum period of time, the Called Node **shall** abort the Physical Link and declare it inactive.
e) The maximum period of time to wait for the first DATA (TYPE 0) C_PDU before aborting a newly activated Physical Link **shall** be a configurable parameter in the implementation.

When a physical link is accepted, SIS is informed with C_PHYSICAL_LINK_MADE. If ALE is not being used the DTS is informed with D_CONNECTION_MADE (this step is done earlier when ALE is used).

Step 3. Caller

The caller follows this procedure when the second procedure in step 1 was followed.

There are two possible outcomes to the protocol for making a physical link: success or failure:

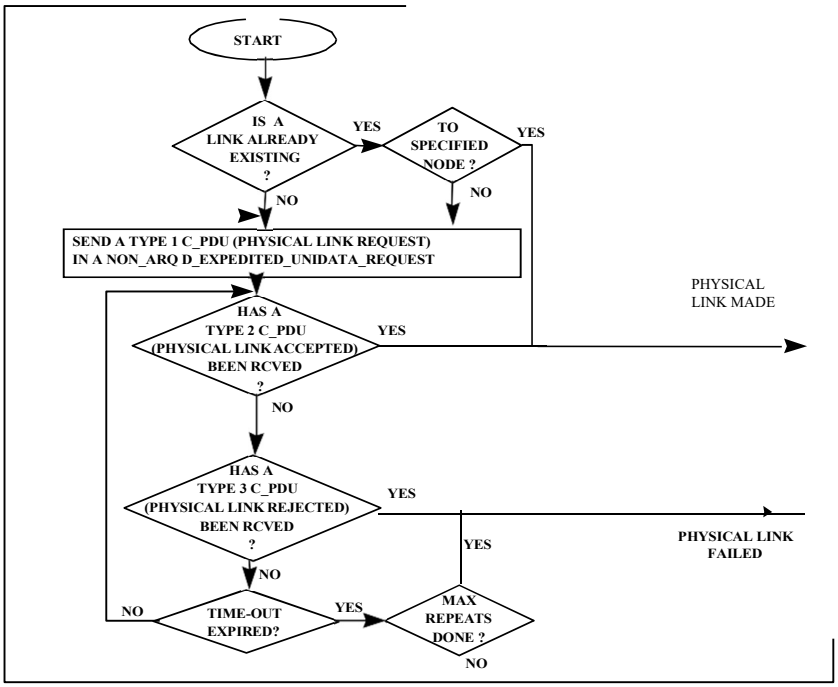
a) Upon receiving a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU, the calling Channel Access Sublayer **shall** proceed as follows:
   (1) the Calling node **shall** declare the Physical Link as *successfully Made*,
   (2) send C_PHYSICAL_LINK_MADE to the SIS and send D_CONNECTION_MADE to the DTS; otherwise,
b) upon receiving a PHYSICAL LINK REJECTED (TYPE 3) C_PDU, the Channel Access Sublayer **shall** declare the Physical Link as *Failed,* and send C_PHYSICAL_LINK_REJECTED to the SIS. otherwise,
c) upon expiration of its timer without any response having been received from the remote node, the Channel Access Sublayer **shall** repeat Step 1 (i.e. send a PHYSICAL LINK REQUEST (TYPE 1) C_PDU and set a response time) and await again a response from the remote node.

The maximum number of times the Caller sends the PHYSICAL LINK REQUEST (TYPE 1) C_PDU without a response from the called node of either kind **shall** be a configurable parameter in the implementation of the protocol.

After having repeated Step 1 the configurable maximum number of times without any response having been received from the remote node, the

Caller's Channel Access Sublayer **shall** declare the protocol to make the physical link as *Failed*, and send C_PHYSICAL_LINK_REJECTED to the SIS.

Figure B-12 and Figure B-13 show the nominal procedures followed by the Caller and Called Channel Access Sublayers for Making a Physical Link, when ALE is not used or after ALE initialization and CAS-1 is used. This STANAG



acknowledges that other implementations may meet the stated requirements.

Figure B-12: Type 1 Channel Access Sublayer protocol for Making a Physical Link (Caller Peer)

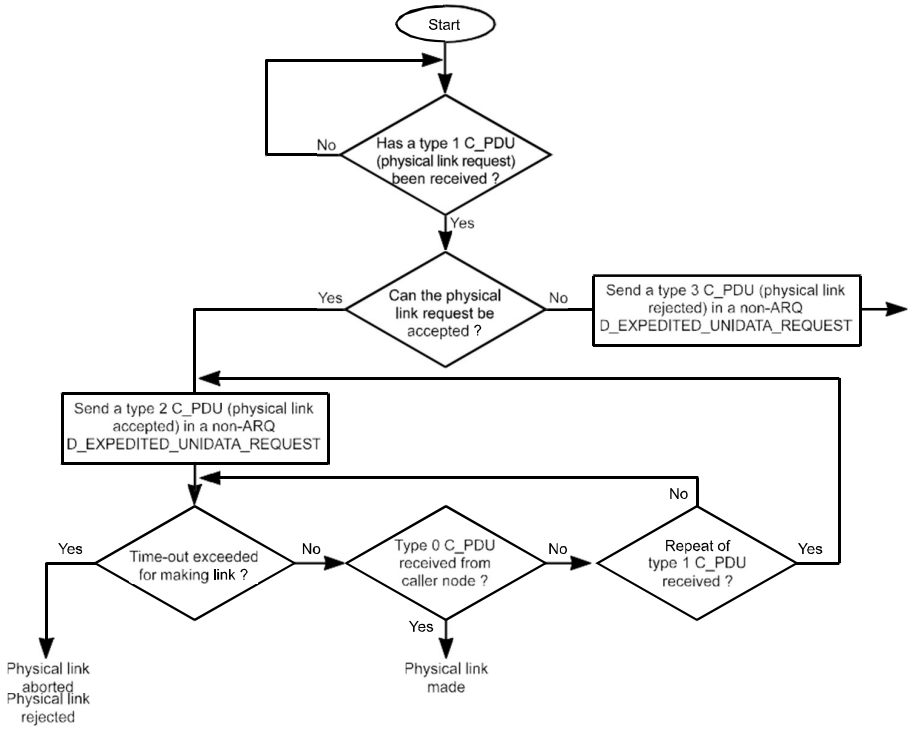


Figure B-13: Type 1 Channel Access Sublayer Protocol for Making a Physical Link (Called Peer)

After having declared the Physical Link as successfully made, the peer Channel Access Sublayers do not carry out any further handshake in order to confirm the successful operation of the Link.

B.7.2.2. Determining Edition 4 Support

The physical link setup is also used to determine support for this edition of the protocol (or subsequent editions). On receipt of a PHYSICAL LINK REQUEST C_PDU with the Ed4 bit set, a node can determine that a peer supports edition 4.

It is likely that an edition 3 (or earlier) implementation receiving a PHYSICAL LINK

REQUEST C_PDU with the Ed4 bit set will ignore this bit and respond with a PHYSICAL LINK ACCEPT C_PDU which does not have the Ed4 bit set. This will enable the local node to determine that the peer does not support edition 4. This mechanism provides a robust way to determine if a peer supports edition 4 (or subsequent).

It is possible, but unlikely, that the edition 3 peer will respond to a C_PDU with the Ed4 bit set by treating it as an error and responding with a WARNING D_PDU. In the event that this happens the local peer **shall** record that the peer does not support edition 4 and **shall** repeat the link establishment without the Ed4 bit set. For subsequent links, the record of edition support can avoid repeating the failed link attempt.

When both peers support edition 4 (or subsequent) the default DTS behavior will be to use the family of D_PDUs with 16 bit Frame Sequence Number. Use of the Short bit allows peers to negotiate use of D_PDUs with 8 bit Frame Sequence Number. This could be beneficial in scenarios where it is known that speeds will be 1200 bps or less, as the D_PDU header overhead is slightly reduced.

B.7.2.3. **Interoperability with Edition 3 Systems using ALE 1:1**

Edition 3 does not specify how to use ALE 1:1, but a number of vendors have implemented a capability. This specification defines use of ALE 1:1 with Ed3, as it will only use Ed3 PDUs to interoperate with and Ed3 system.

The choice of mode (IMPLICIT CAS-1 vs EXPLICIT CAS-1) should be made based on how the Ed3 system works. It is known that there are Ed3 systems which used both modes, although it is believed that EXPLICIT CAS-1 is more common.

B.7.2.4. **Clean Close State**

Prior to terminating an ALE Link or a physical link, it is desirable that the CAS 1 state is “clean”, with no pending traffic in either direction. This maximizes efficiency and minimizes errors.

A node is in a clean state when:

a. It has no data sent by the peer to acknowledge; and
b. It has no data sent that has not been acknowledged

A node will naturally reach this state when a link goes idle and there is no data left to be transferred. A model of “close links when there is nothing to transfer” can be provided cleanly.

NOTE: A node can only determine if it is in clean state when it is the nodes turn to transmit, typically immediately after it has received data from the peer.

Clean Close State is only needed when a link is closed where a different link may be opened after the link is closed. If an ALE link is being closed for re-linking with expectation of resuming the current link, there is no need to reach Clean Close State, as the procedure set out in this section B.7.2.9 can be followed to ensure no loss.

In order to support requests to close busy links, a mechanism to get links into a clean state is provided. When a request to close a link is received, the procedure described in this section **shall** be used to reach a clean close state with peers that support this version of the protocol. This procedure **shall not** be used with Edition 3 peers.

If the node is not in a clean close state, the requirements are to finish transfer of active traffic on the link and to cease using the link for new traffic. In order to reach clean close state, a PHYSICAL LINK BREAK REQUEST C_PDU is sent, which contains the reason for closing the link. This message is sent using ARQ and so does not need to be timed or repeated.

Once a node has sent a PHYSICAL LINK BREAK REQUEST C_PDU it will monitor link state and **shall** initiate closing the link as soon as the node can determine that it is in clean close state, which will usually be after receiving data from the peer.

When a node receives a PHYSICAL_LINK_BREAK_REQUEST, its actions will depend on the Reason provided.

Where the reason is “Channel Share”, or “Operator Request”, the node **shall** take actions to move the link to clean close state. No new Data C_PDUs **shall** start transmission apart from client confirmations, noting that this control may need interaction with the DTS. Where a Data C_PDU has started transmission (i.e., some D_PDUs have been sent) then the rest of the C_PDU **shall** be transmitted. Note that reaching clean close state may take a number of transmissions in each direction. The node **shall** monitor link state and **shall** initiate closing the link as soon as the node reaches clean close state.

The “No Data” reason is advisory. It the node receiving this has data to transmit, it **shall** continue to use the link. The node **shall** monitor link state and **shall** initiate closing the link as soon as the node reaches clean close state.

Action on the “Higher Priority” reason depends on the priority. New C_PDUs of lower priority **shall not** start transmission. However, new C__PDUs of the same or higher priority **shall** start transmission. The node **shall** monitor link state and **shall** initiate closing the link as soon as the node reaches clean close state.

B.7.2.5. Protocol for Terminating an ALE Link (IMPLICIT CAS-1)

When ALE is used in IMPLICIT CAS-1 mode, the following procedure **shall** be used for terminating the ALE link. The procedure also closes all physical links associated with the ALE link.

When ALE is used, one of the nodes will terminate the link using ALE_LINK_TERMINATE and the ALE layer will communicate termination to other nodes, where the CAS layer will receive ALE_LINK_TERMINATED.

ALE links should generally be terminated promptly, either when there is no data to send or when SIS wishes to schedule transfer of data to a different peer which will require a new ALE link and a new physical link.

If C_PHYSICAL LINK BREAK is received from SIS, the Clean Close Procedure specified in Section B.7.2.4 **shall** be followed using the reason provided by the C_PHYSICAL_LINK BREAK.

Then the ALE Link Break procedure is started, which will terminate the ALE link for all nodes.

Similarly, if ALE analysis determines that the link should be broken (due either to link quality being poorer than anticipated or to the length of time), the Clean Close Procedure specified in Section B.7.2.4 **shall** be followed using the appropriate ALE reason. Then the ALE Link Break procedure is started, which will terminate the ALE link for all nodes.

Otherwise the ALE Link Break procedure shall be started when any of the following conditions apply:

1. For each physical link, the node is in Clean Close State as described in Section B.7.2.4 or DTS has reported link failure with D_CONNECTION_LOST; and
2. If any Non-ARQ data has been sent or received over the link, that a configurable timer since last non-ARQ data was sent or received has expired; and
3. If the ALE link is multi-node, that no traffic between other nodes has been detected more recently than a configurable time in the past.

In addition to this there **shall** be a configurable ALE Link timer. If no data has been received over the ALE link (or detected between other nodes in a multi-node link) within this timer, then the ALE Link Break procedure **shall** be followed.

The ALE Link Break procedure is for a node to use ALE_LINK_TERMINATE whenever the preceding conditions are met. Any physical links associated with the ALE link **shall** be declared broken, by sending D_CONNECTION_TERMINATED to the DTS.

If the link was in Clean Close state on termination, then C_PHYSICAL_LINK_BROKEN **shall** be sent to the SIS.

If the link was non in Clean Close state, which may be due to DTS initiated Link Closure or ALE timeout, then C_PHYSICAL_LINK_BROKEN **may** be sent the SIS. Alternatively, another ALE link **may** be initiated following the protocol for resuming a physical link, as specified in Section B.7.2.9.

If an ALE_LINK_TERMINATED is received at any time, the ALE link has been terminated by a peer or a local operator. A peer is expected only to terminate an ALE link when in Clean Close state. Any physical links associated with the ALE link **shall** be declared broken, by sending D_CONNECTION_TERMINATED to the DTS and C_PHYSICAL_LINK_BROKEN to the SIS.

B.7.2.6. Protocol for Terminating an ALE Link (EXPLICIT CAS-1)

When ALE is used in EXPLICIT CAS-1 mode, the following procedure **shall** be used for terminating the ALE link.

The procedure specified in Section B.7.2.8is followed to break the CAS-1 link.

After the CAS-1 link is closed, the ALE link is terminated by the initiator when it receives PHYSICAL LINK BREAK CONFIRM. The ALE link is closed using ALE_LINK_TERMINATE.

B.7.2.7. Protocol for ALE Relinking

There are situations where it is desired to continue a physical link, but to relink ALE. This might be because the performance of the current link is too poor or because the current link has been running for an extended period, and it is desired to determine if a better link can be achieved.

In EXPLIT CAS-1 mode, ALE may be relinked at any time by either node. This can be done using ALE_LINK_RELINK, or using ALE_LINK_TERMINATE immediately followed by ALE_LINK_MAKE.

The following procedure is used in IMPLICIT CAS-1 mode.

This procedure **shall not** be used with Edition 3 peers.

When a node determines a desire to relink, it **shall** send a ALE RELINK C_PDU with value of either “ALE Quality” or “ALE Time” dependent on reason for relinking. The node **shall** then wait for the DTS acknowledgement of the ALE RELINK C_PDU, to ensure that the peer is aware that the relink procedure is being initiated. The node **shall** then immediately relink at ALE level. If the ALE supports ALE Link Management (ALM), the link **shall** be relinked using ALE_LINK_RELINK. Otherwise the current link is terminated using ALE_LINK_TERMINATE and then a new link made immediately using ALE_LINK_MAKE.

Following relinking, the node performing the relinking **shall** initiate the state resumption procedure specified in Section B.7.2.9.

On receiving an ALE RELINK C_PDU, a node **shall** send DTS acknowledgement of the C_PDU and **shall not** send any other data. It will then wait for the peer to continue the relinking procedure. This node **may** be notified of relinking using ALM. Alternatively it may receive ALE_LINK_TERMINATED. In the latter case, it **shall not** initiate any new ALE requests for a configurable period, but **shall** wait for an incoming ALE_LINK_REMOTE. If the incoming request is not from the relinking peer and there is a mechanism to reject the incoming call, the node **may** do so.

B.7.2.8. Protocol for Breaking a Physical Link

In the specification that follows of the protocol for breaking a physical link, the node which requests the breaking of the Physical Link will be referred to as the Initiator or initiating node, while the node which receives the request will be referred to as the Responder or responding node. This protocol is used when ALE is not used or in EXPLICIT CAS-1 mode.

Physical Links will be broken in the following situations:

1. If C_PHYSICAL LINK BREAK is received from SIS; or
2. D_CONNECTION_LOST reported by the DTS;.

If D_CONNECTION_LOST is reported by the DTS, the SIS is informed with C_PHYSICAL_LINK_BROKEN.

For C_PHYSICAL_LINK BREAK, the Clean Close Procedure specified in Section B.7.2.4shall be followed using the reason provided by the C_PHYSICAL_LINK BREAK.

It is recommended that SIS client flow control is used to avoid large queue build up in the DTS. However, there needs to be some measure of queueing in DTS in order for the DTS to operate efficiently. When a link is broken prior to all traffic being sent, it is generally desirable to close the link quickly. For this reason, link break procedure makes use of expedited data, which will ensure that the link break protocol is not blocked behind other traffic on the link.

The protocol for breaking the Physical Link **shall** consist of the following steps:

Step 1: Initiator

a) To start the protocol, the Initiator's Channel Access Sublayer **shall** send a type 4 C_PDU (PHYSICAL LINK BREAK).
b) Upon sending this C_PDU the Channel Access Sublayer **shall** start a timer which is set to a value greater than or equal to the maximum time required by the Called Node to send its response (this time depends on the modem parameters, number of re-transmissions, etc.).

Step 2: Responder

a) Upon receiving the type 4 C_PDU the Responder's Channel Access Sublayer **shall** declare the Physical link as *broken* and send a PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU. Then D_CONNECTION_TERMINATED is sent to the DTS and C_PHYSICAL_LINK_BROKEN is sent to the SIS.

Step 3: Initiator

a) Upon receiving a PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU, the Initiator's Channel Access Sublayer **shall** declare the Physical Link as *broken by sending D_CONNECTION_TERMINATED to the DTS and C_PHYSICAL_LINK_BROKEN to the SIS.*.

b) Upon expiration of its timer without any response having been received from the remote node, the Initiator's Channel Access Sublayer **shall** repeat step 1 and wait again for a response from the remote node.

c) After having repeated Step 1 a maximum number of times (left as a configuration parameter) without any response having been received from the remote node, the Initiator's Channel Access Sublayer **shall** declare the Physical Link as *broken by sending D_CONNECTION_TERMINATED to the DTS and C_PHYSICAL_LINK_BROKEN to the SIS.*.

Figure B-14(a) and (b) show the procedures followed by the Initiator and Responding Channel Access Sublayers for Breaking a Physical Link.

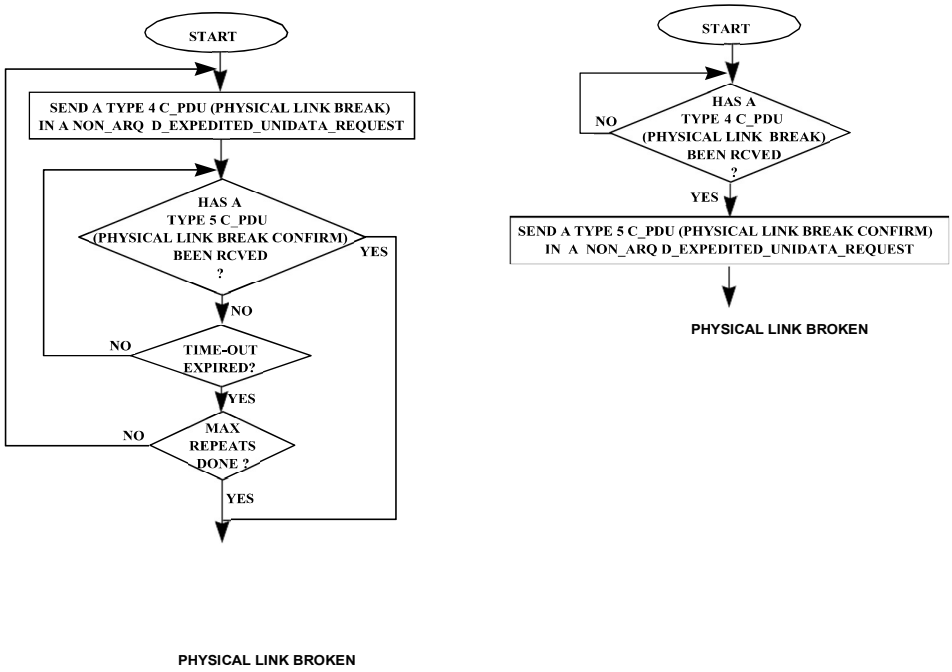


Figure B-14: Type 1 Channel Access Sublayer protocol for Breaking a Physical Link
(a) Initiator Peer. (b) Responding Peer.

B.7.2.9. Protocol for Resuming a Physical Link

The process **may** be used with Edition 4 peers when an ALE link has failed. In this situation, it is likely that both peers will be aware that the ALE link has failed. This process allows both peers to continue without reset of CAS-1 status. This process **shall not** be used with Edition 3 peers.

For this process to work, both nodes need to have preserved CAS-1 / ARQ status from the previous ALE link.

The node initiating this process shall send a PHYSICAL LINK RESUME C_PDU using the Expedited ARQ service and wait for a response.

A node receiving a PHYSICAL LINK RESUME C_PDU **may** accept the resumption if it has preserved CAS-1 state since the previous ALE link, which it does by sending a PHYSICAL LINK RESUME ACCEPT C_PDU using the Expedited ARQ service. It then continues operation with standard protocol, and waits for the ALE Link initiator to transmit data on the ALE link.

The receiving node **may** also reject the request by sending a PHYSICAL LINK RESUME REJECT C_PDU using the Expedited ARQ service. The receiver will

then proceed as for reception of a new ALE link,

When the initiator receives a PHYSICAL LINK RESUME ACCEPT C_PDU, it will then proceed with data transfer following normal procedures starting with the preserved CAS-1 state.

When the initiator receives a PHYSICAL LINK RESUME REJECT C_PDU, it will then reset local CAS-1 and ARQ status, and the proceed as for a newly initiated ALE link.

In the event that a node which has preserved CAS-1 status receives an ALE link and the first data received is not a PHYSICAL LINK RESUME PDU, the ARQ status is reset.

B.8. Protocol for Exchanging Data C_PDUs.

The protocol for exchanging C_PDUs between peers is simple and straightforward, since the Channel Access Sublayer relies entirely on its supporting (i.e, Data Transfer) sublayer to provide delivery services.

The sending peer **shall** accept S_PDUs from the Subnetwork Interface Sublayer, envelop them in a “DATA” C_PDU (by adding the C_PCI) and send them to its receiving peer via its interface to the Data Transfer Sublayer.

The receiving peer **shall** receive DATA C_PDUs from the Data Transfer Sublayer interface, check them for validity, strip off the C_PCI, and deliver the enveloped S_PDU to the Subnetwork Interface Sublayer.

B.9. Deprecated Service

The Hard Links service specified in Edition 3 are optional in this edition and deprecated. It is anticipated that this services will be removed in future updates of this specification. This 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.

B.9.1. Hard Links

This optional deprecated service is defined as the HARD LINKS profile option.

Hard Links are specified in Edition 3 Annex A in a manner that does not conflict with any of the Edition 4 specification. So use of the Edition 3 specification is

straightforward. Section B.3.2 specifies how to establish a physical link which may either be a hard link or a soft link.

B.10. Changes in Edition 4

This annex has the following changes relative to Edition 3.

1. Explicit setting out of single peers access vs multiple peer access as part of the channel model. This was implicit in Edition 3. It needs to be made explicit in order to clearly document ALE and Duplex operation.

2. Use of 1:1 ALE for connecting to a peer is documented explicitly.

3. The model of duplex operation is set out, both with and without ALE.

4. Where ALE is not used, the specification requires CAS-1 linking. The generic text around handling ALE-like systems is removed.

5. The CAS-1 negotiation is extended to determine if the peer supports Edition 4, and if so to negotiate the choice of D_PDUs to use on the soft link.

6. Some new Error types are introduced, arising from ALE and Duplex support.

7. C_CHANNEL_AVAILABILITY service primitive is removed. It is listed, but not defined. It does not appear to be useful.

8. Errors for C_ primitives explicitly listed.

9. Support for hard links is optional and deprecated.

10. Incorporation of notes from STANAG 5066 Edition 3 Annex H.

11. Removal of handling Expedited Data is optional and deprecated, in line with Annex A.

12. Add ALE Link Termination Procedure

13. Add support for ALM and Relinking

14. Added EXPLICIT CAS-1 and IMPLICIT CAS-1 modes

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.

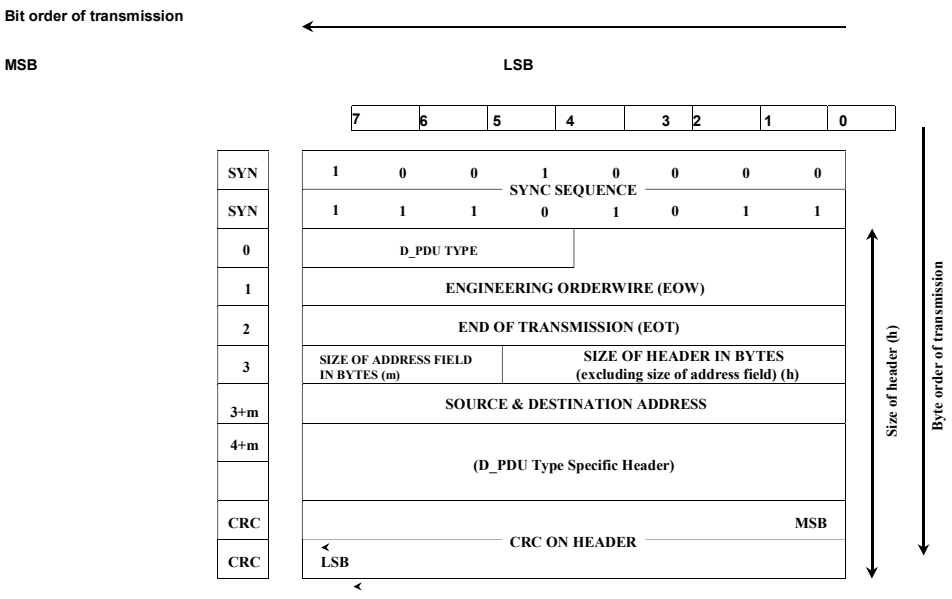


Figure C-2 (a). Generic D_PDU C-Frame Structure.

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.

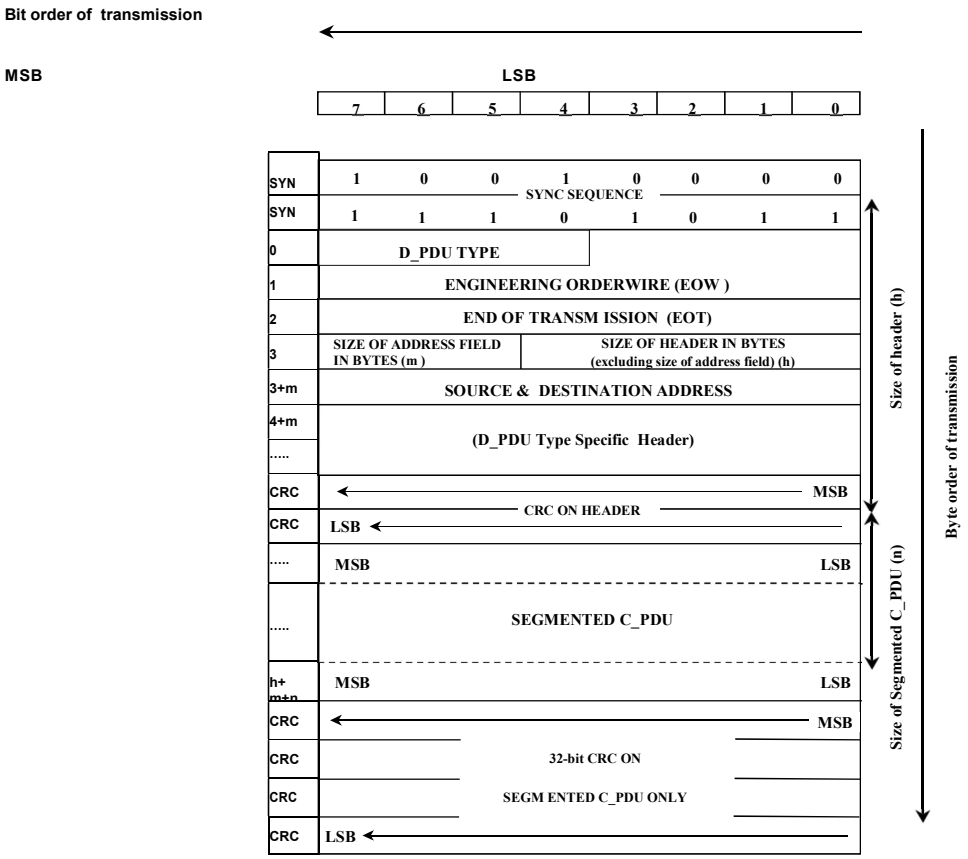


Figure C-2 (b). Generic D_PDU I and I+C Frame Structure.

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.

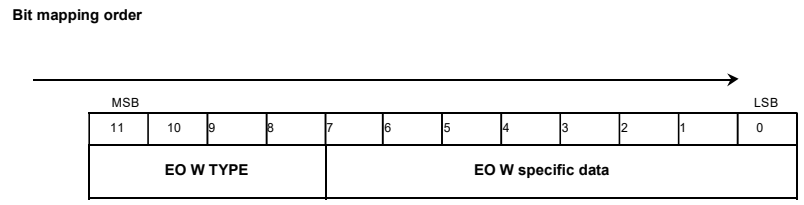


Figure C-3 (a). Generic EOW message.

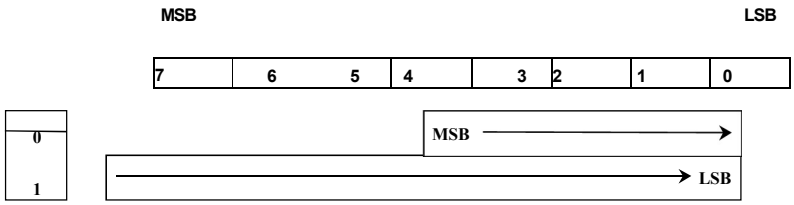


Figure C-3 (b). EOW mapping convention in D_PDU header.

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).

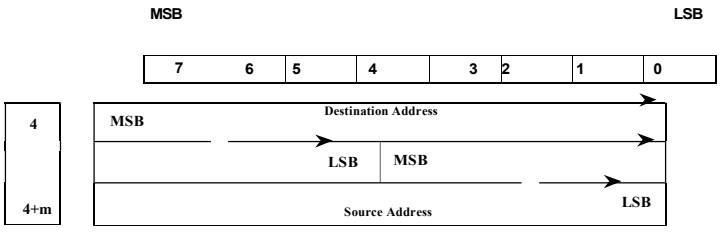


Figure C-4 (a). Address mapping convention in D_PDU header, assuming address-field size is odd

.

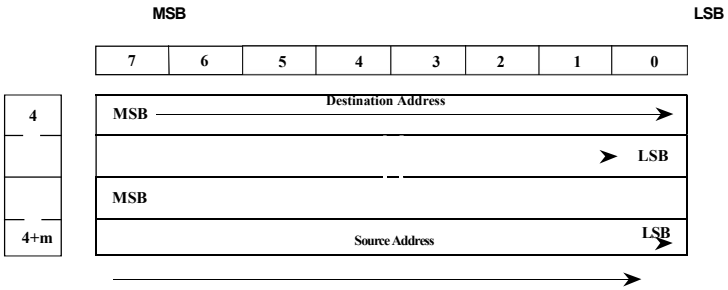


Figure C-4(b). Address mapping convention in D_PDU header, assuming address-field size is even.

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.

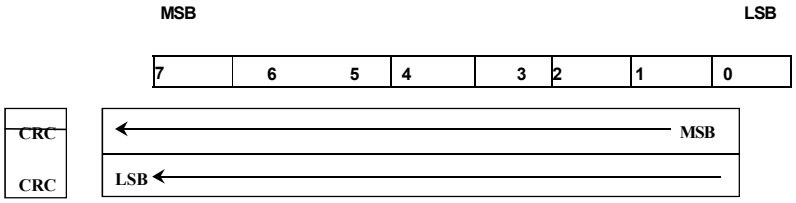


Figure C-5. CRC mapping convention in D_PDU header.

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.

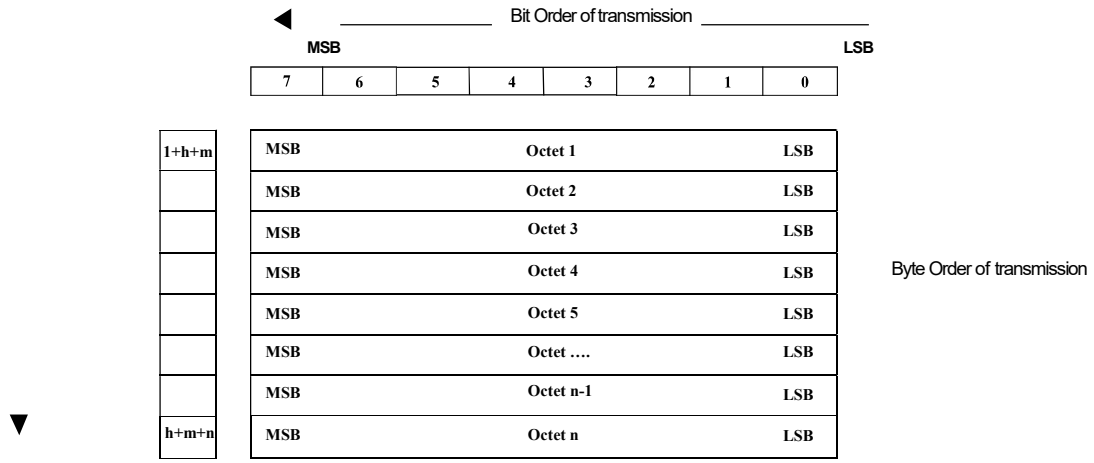


Figure C-6. Segmented C_PDU mapping convention in D_PDU structure.

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.

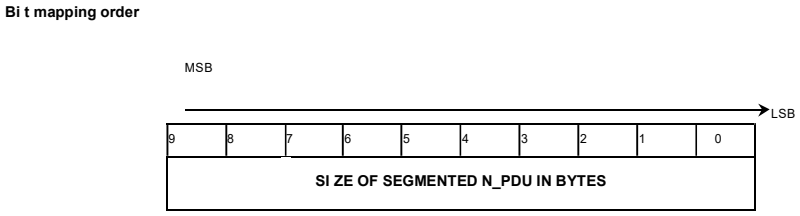


Figure C-7. SIZE OF SEGMENTED C_PDU Field.

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.

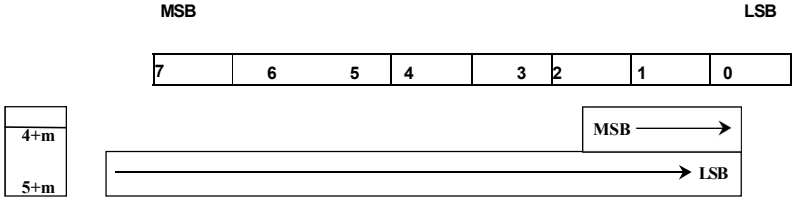


Figure C-8. Segmented C_PDU Size mapping convention in all applicable D_PDU header structures.

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

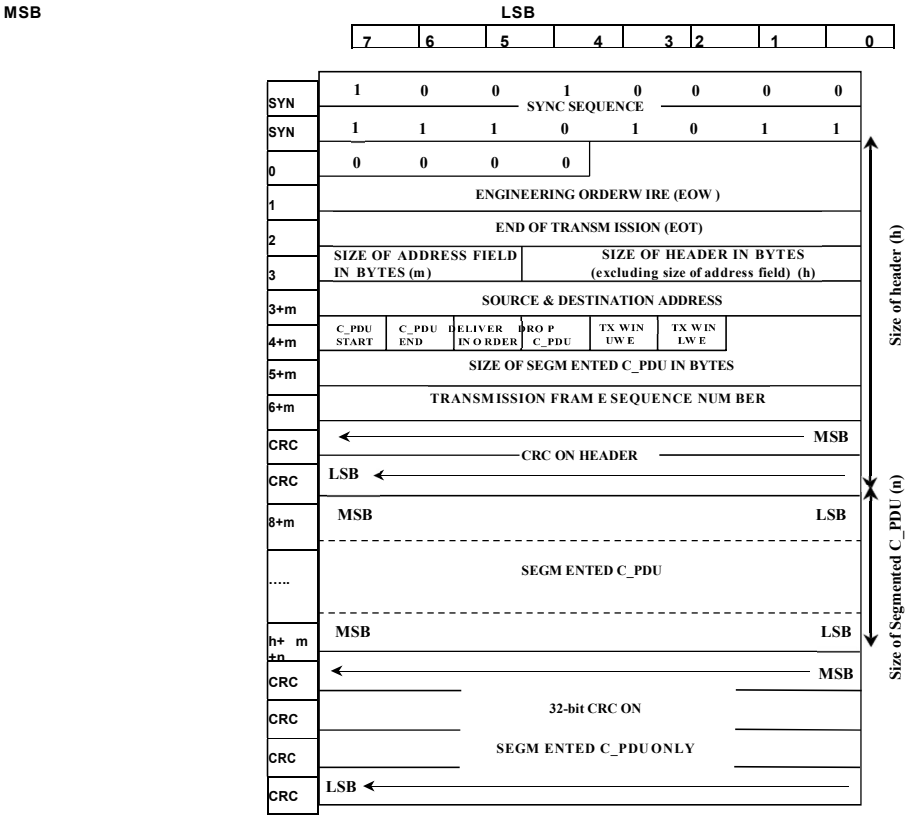


Figure C-9. Frame format for ED3-DATA-ONLY D_PDU Type 0

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

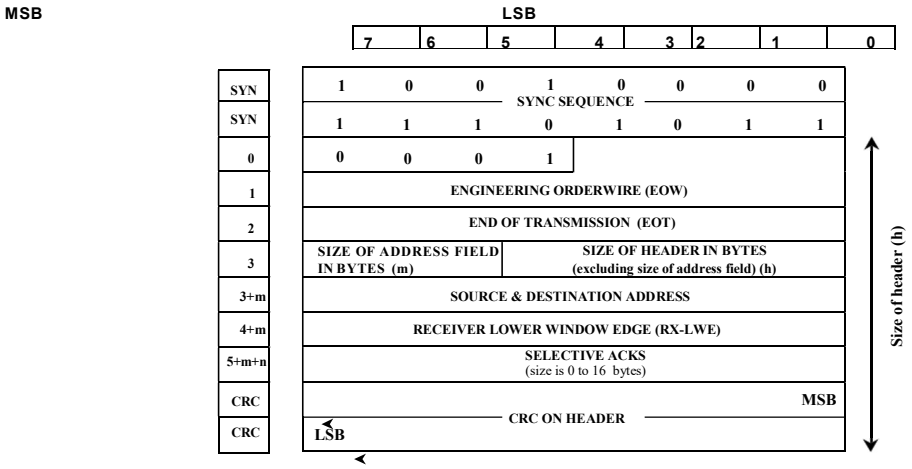


Figure C-11. Frame format for ED3-ACK-ONLY D_PDU Type 1

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

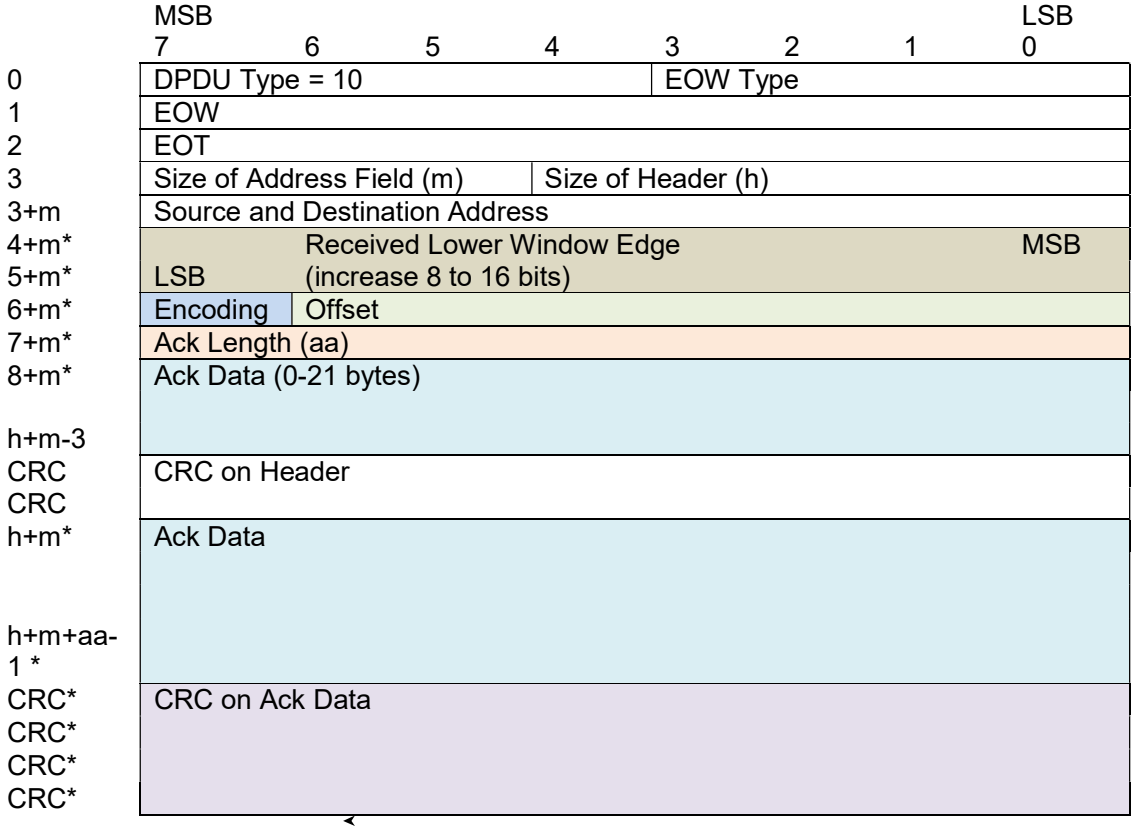


Figure C-12. Frame format for ACK-ONLY D_PDU Type 10

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.

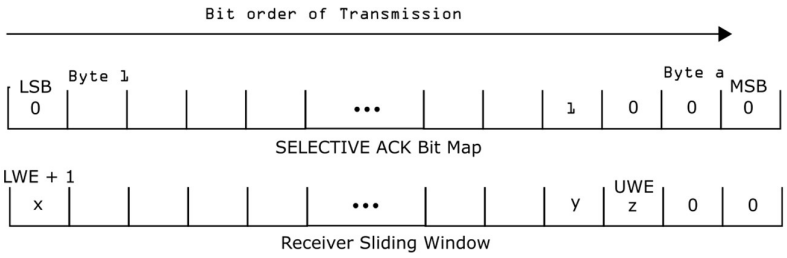


Figure C-13. Constructing the SELECTIVE ACK Field
(Note: the bits that are set (1) and cleared (0) are representing example bits)

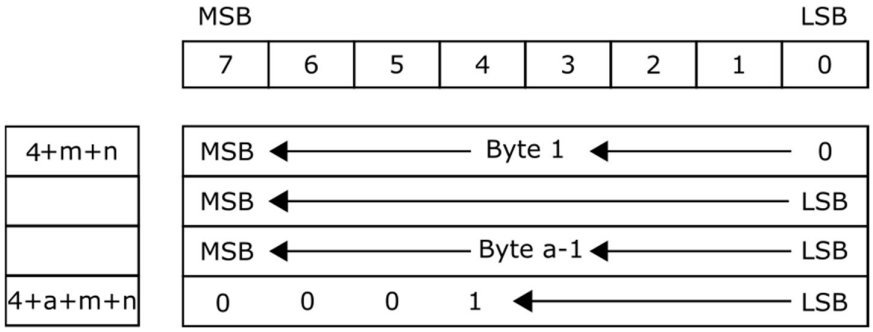


Figure C-14. SELECTIVE ACK mapping convention
(Note: the example bits that were shown in Figure C-13 can been seen in this mapping example)

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.

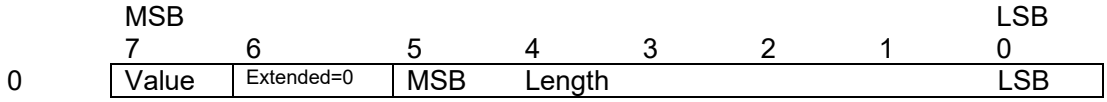


Figure C-15. Run Length Encoding: Single Byte Encoding

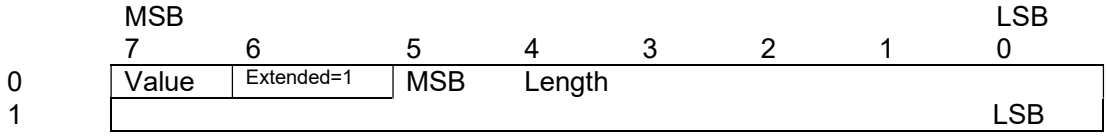


Figure C-16. Run Length Encoding: Two Byte Encoding

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

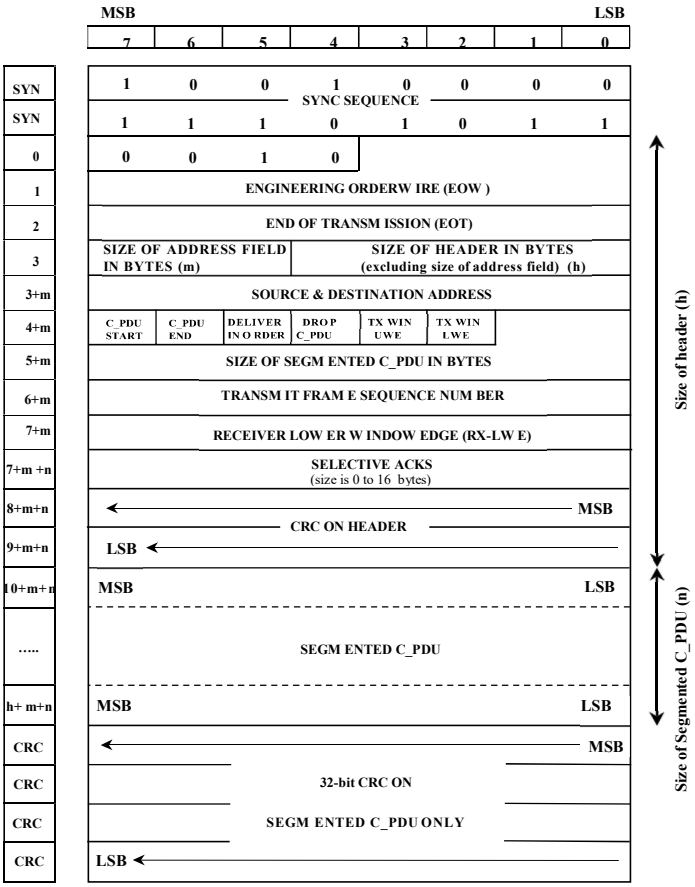


Figure C-17. Frame format for ED3-DATA-ACK D_PDU Type 2

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

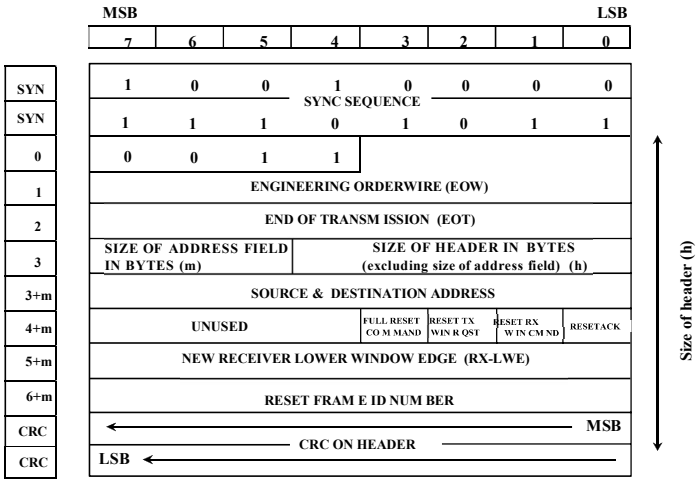


Figure C-19. Frame format for ED3-RESET/WIN-RESYNC D_PDU Type 3

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**

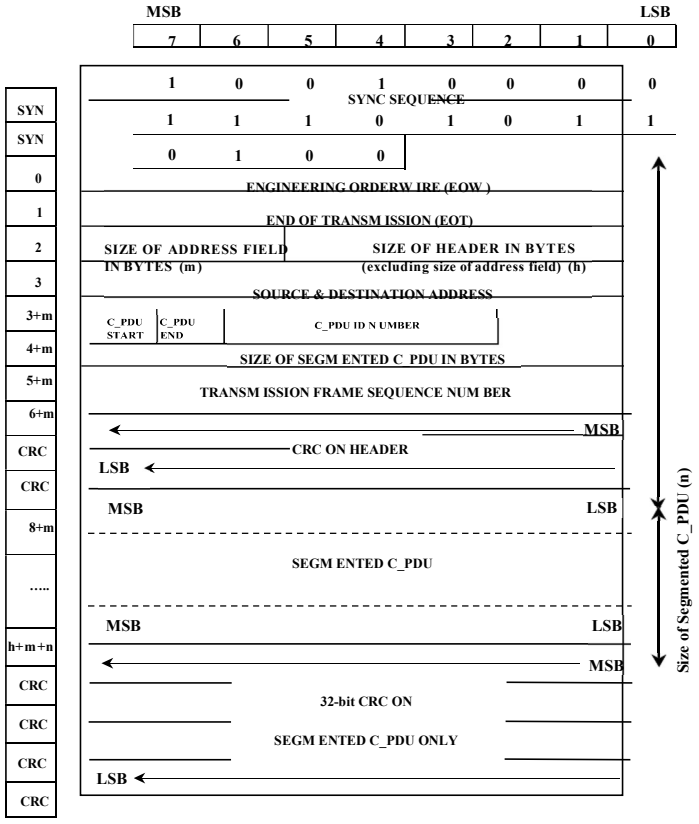


Figure C-21. Frame format for EXPEDITED DATA ONLY D_PDU Type 4

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)**

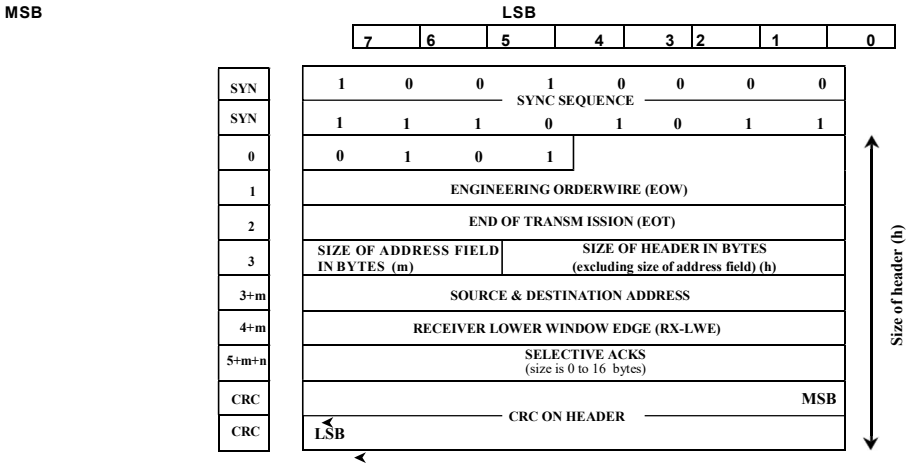


Figure C-22. Frame format for EXPEDITEDACK-ONLY D_PDU Type 5

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)**

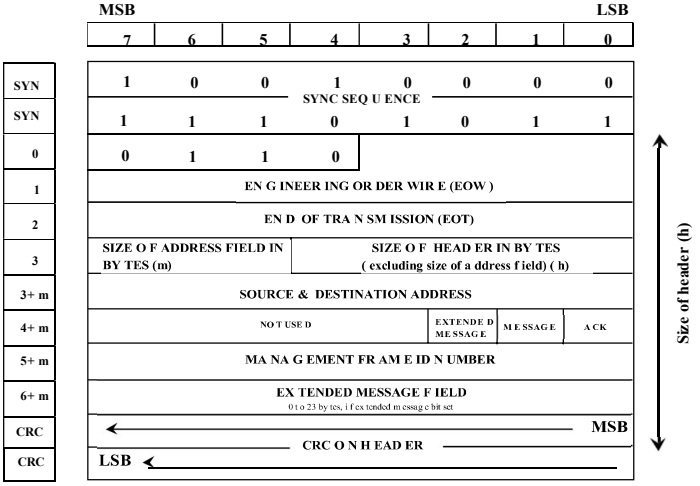


Figure C-23. Header format for MANAGEMENT D_PDU TYPE 6

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)**

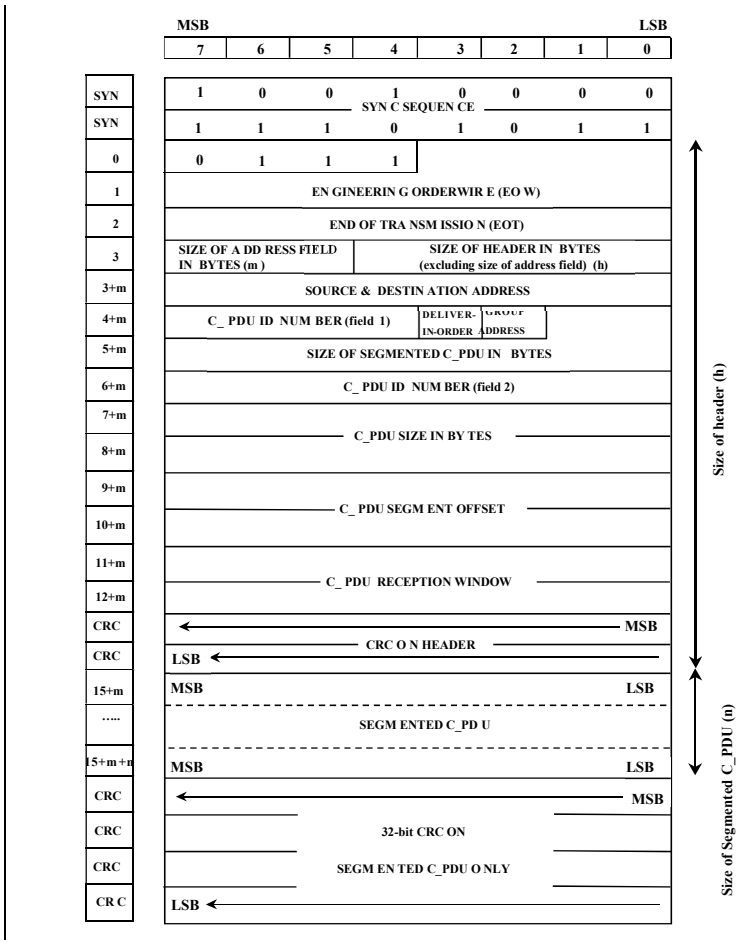


Figure C-24. Frame format for NON-ARQ-DATA D_PDU TYPE 7

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.

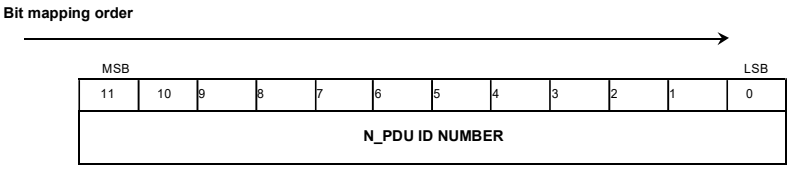


Figure C-25. C_PDU ID NUMBER Field

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).

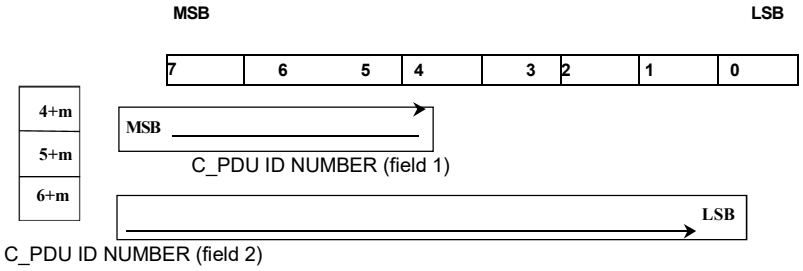


Figure C-26. C_PDU ID NUMBER mapping convention in D_PDU header

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.

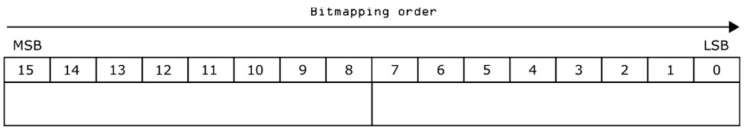


Figure C-27. C_PDU SIZE Field

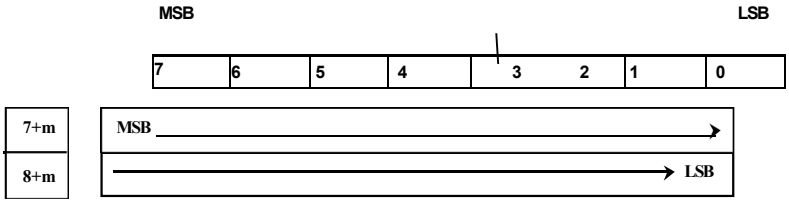


Figure C-28. C_PDU SIZE mapping convention in D_PDU header

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.

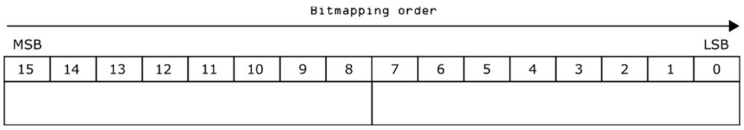


Figure C-29. C_PDU SEGMENT OFFSET Field

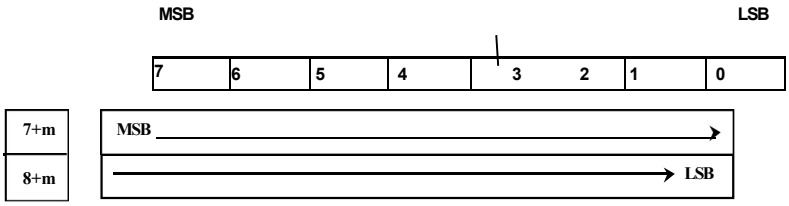


Figure C-30. C_PDU SEGMENT OFFSET mapping convention in D_PDU header

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.

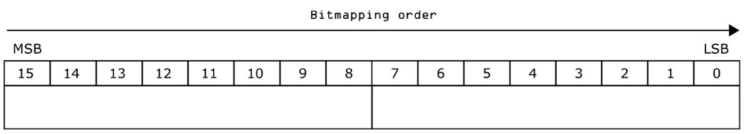


Figure C-31. C_PDU RECEPTION WINDOW Field

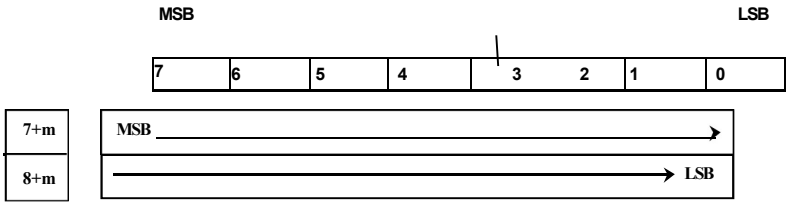


Figure C-32. C_PDU RECEPTION WINDOW mapping convention in D_PDU header

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

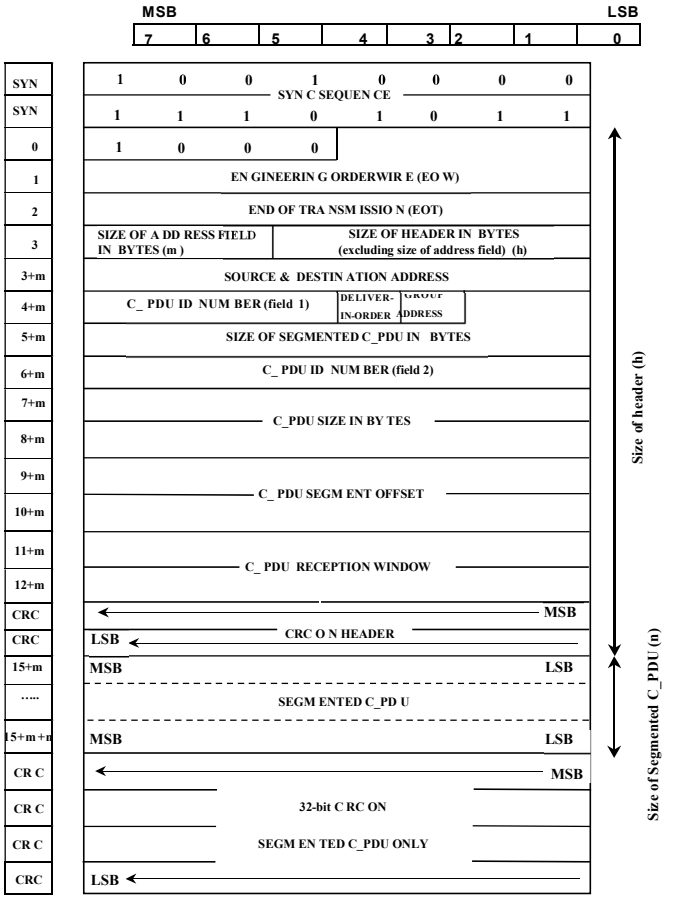


Figure C-33. Frame format for EXPEDITED-NON-ARQ DATA D_PDU Type 8

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**

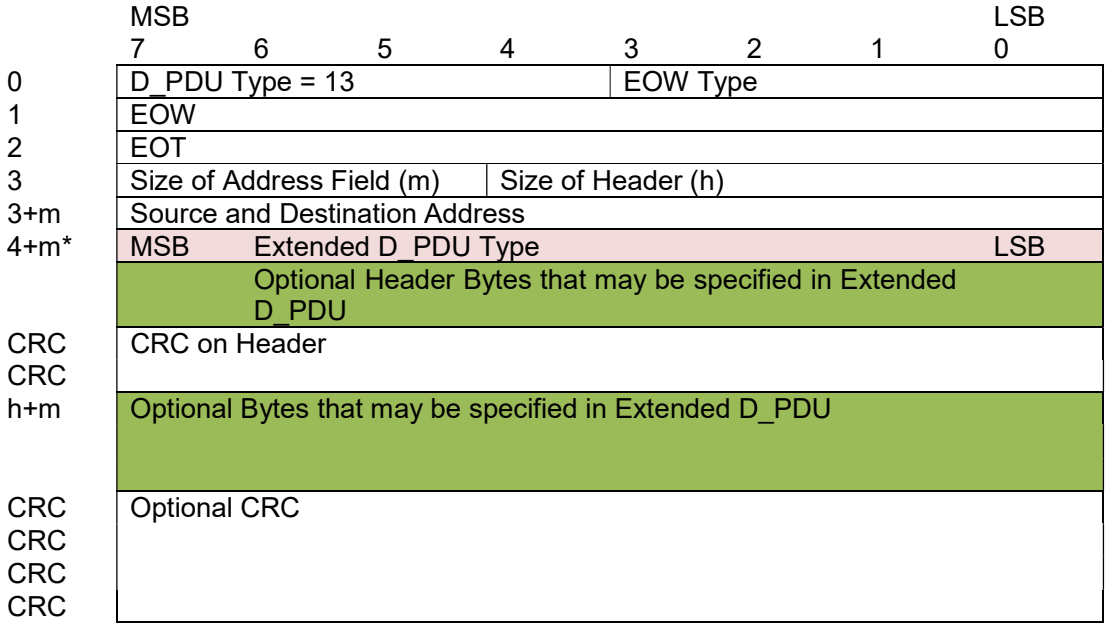


Figure C-34. Frame format for EXTENSION D_PDU Type 13

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.

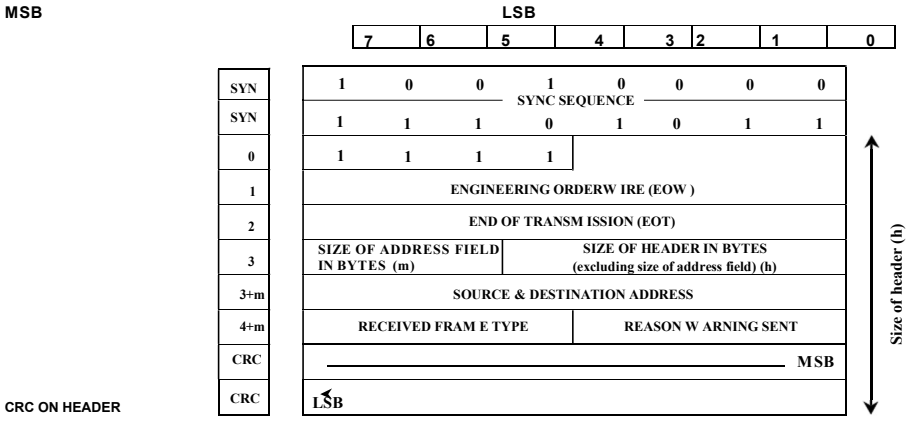


Figure C-36. Frame format for WARNING D_PDU Type 15

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.

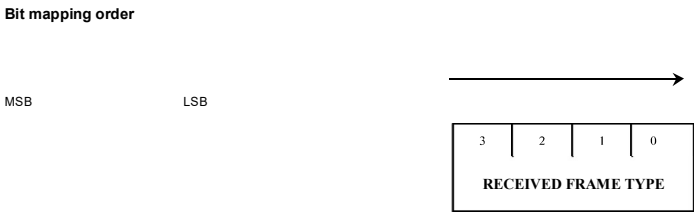


Figure C-37. RECEIVED FRAME TYPE Field

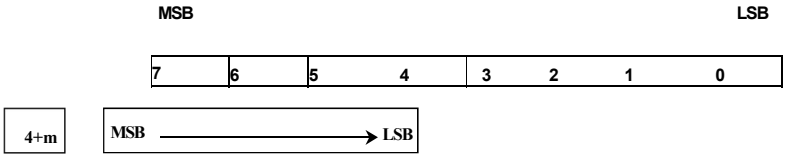


Figure C-38. RECEIVED FRAME TYPE mapping convention in D_PDU header

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.

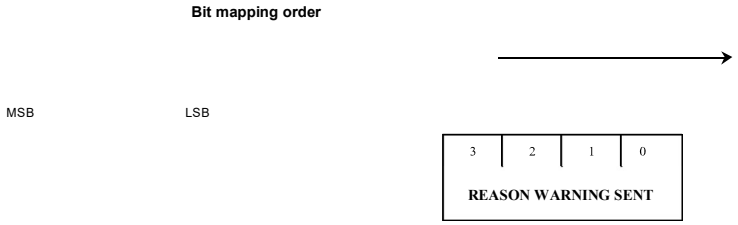


Figure C-39. REASON WARNING SENT Field

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

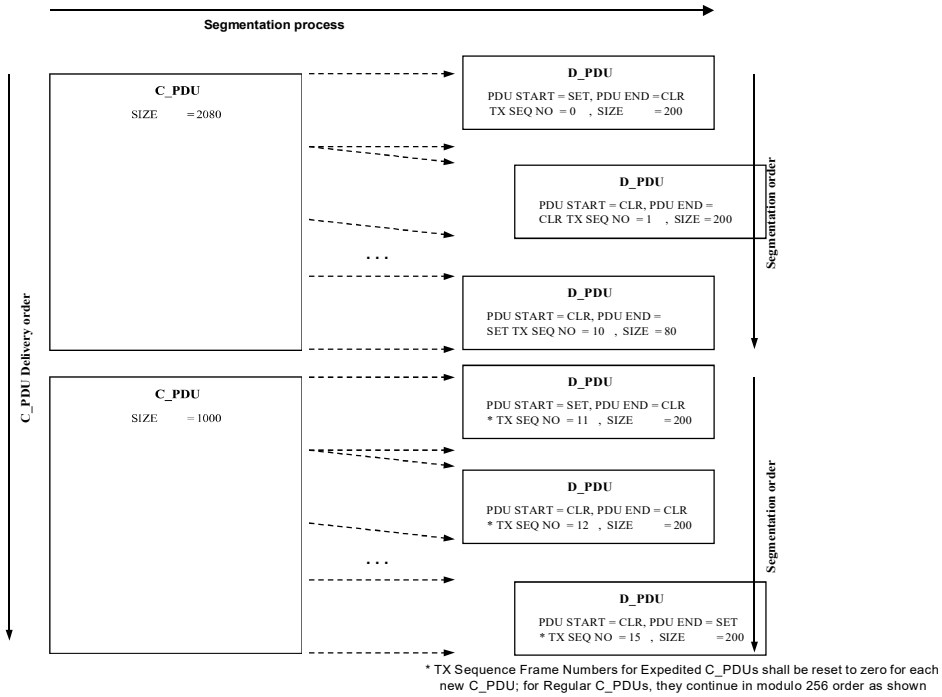


Figure C-41. C_PDU Segmentation for ARQ Delivery Services (Regular and Expedited Service)

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.]

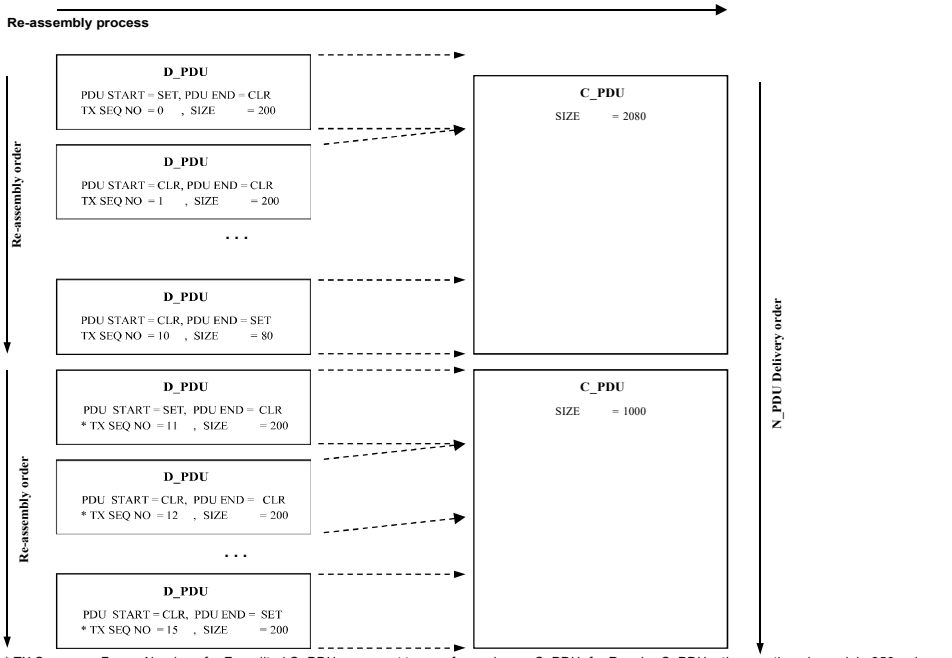


Figure C-42. C_PDU Re-assembly for ARQ Delivery Services (Regular and Expedited Service)

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**

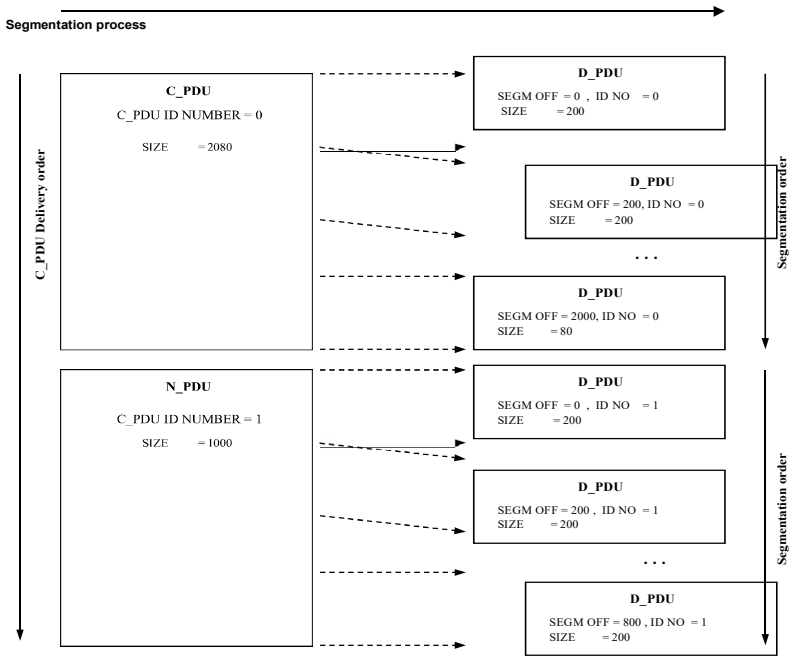


Figure C-43. C_PDU Segmentation for Non-ARQ Delivery Services (Regular and Expedited Service)

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.

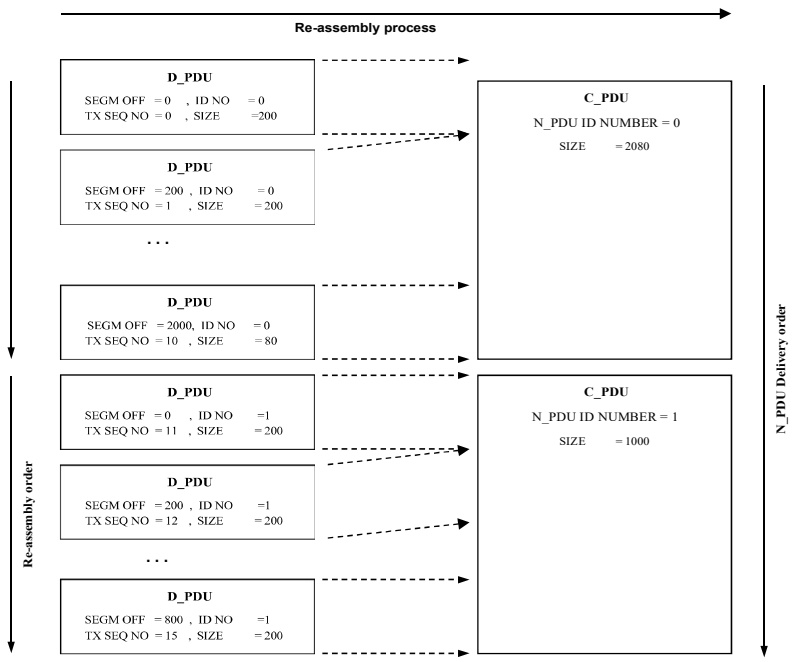


Figure C-44. C_PDU Re-assembly for Non-ARQ Delivery Services (Regular and Expedited Service)

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.

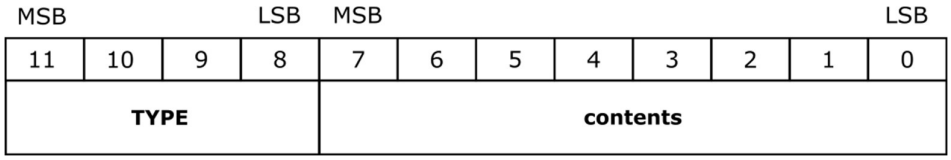


Figure C-45. Format of EOW Messages

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

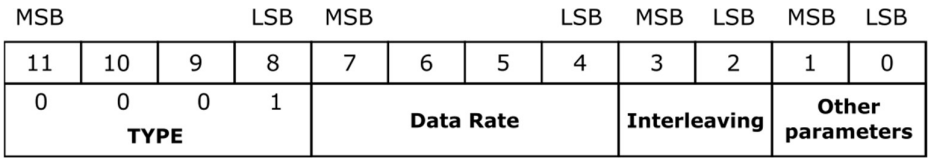


Figure C-46. Format of ED3-BASIC-RATE EOW Message.

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**

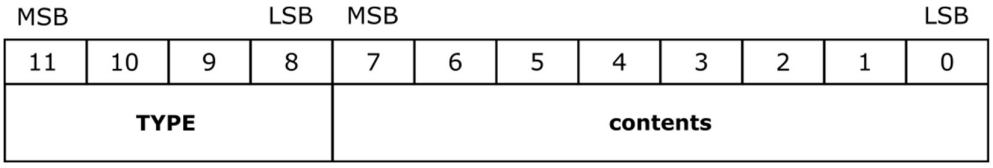


Figure C-48. Format of ED3-CAPABILITY EOW Message

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)

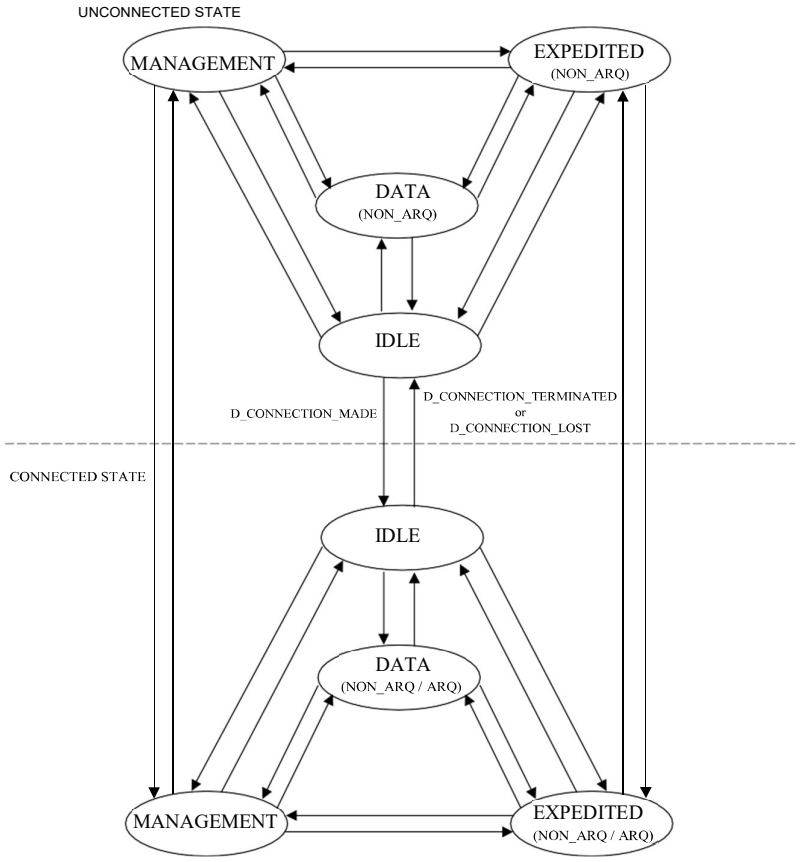


Figure C-54. Nominal State and Transition Diagram (Data Transfer Sublayer)

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.

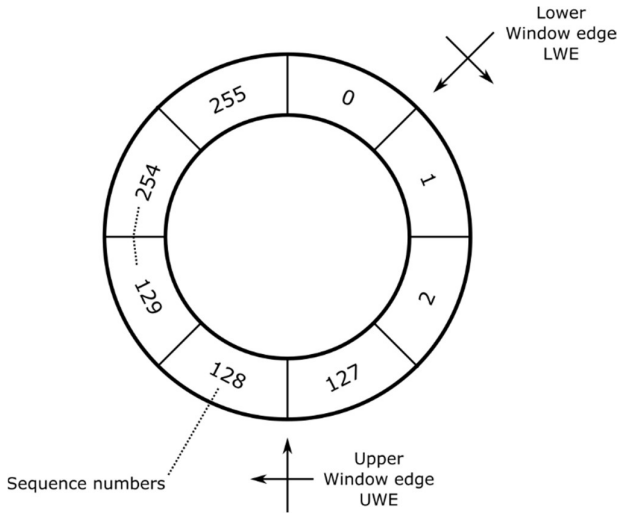


Figure C-55. Sequence-number Space (8 bit): Integers 0..255

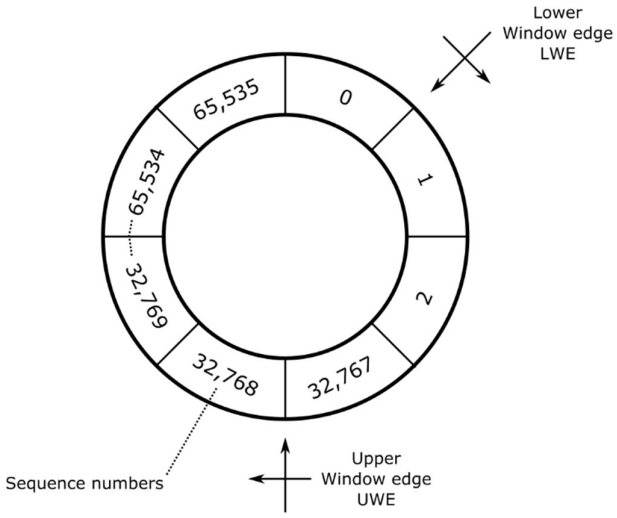


Figure C-56. Sequence-number Space (16 bit): Integers 0..65,535

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.

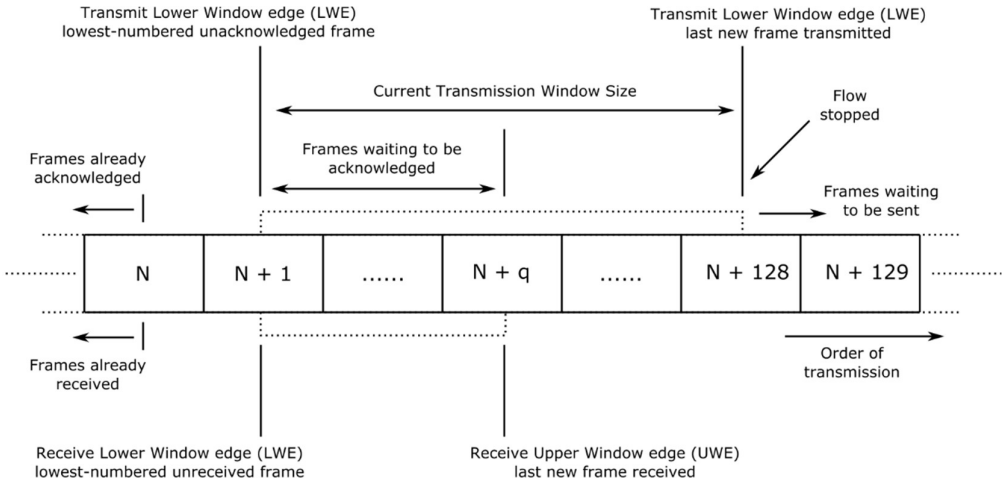
o) If the value of the RX LWE field in any ACK Type 1 or DATA-ACK Type 2 D_PDU is greater than the TX LWE, the Data Transfer Sublayer **shall** (16) declare all D_PDUs with Frame Sequence Numbers between the TX LWE and the RX LWE value as acknowledged, and **shall** (17) advance the TX LWE by setting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

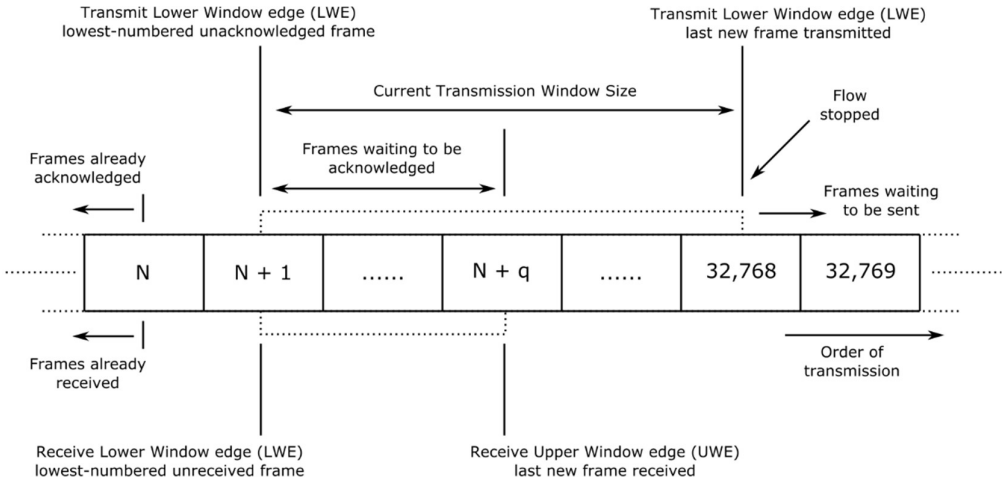
[Note that, although the flow control limitations limit the number of D_PDUs to be transmitted in one transmission, other protocol parameters will also effect this in an indirect way; the structure of the EOT parameter allows a maximum transmission interval of about 2 minutes. If a D_PDU size of 200 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.

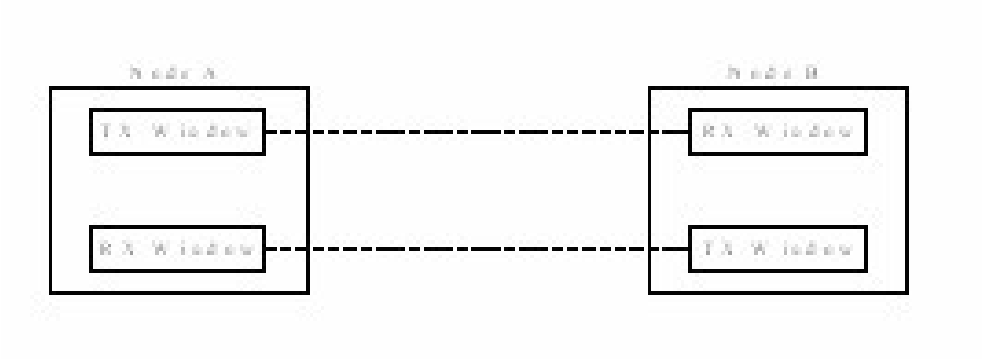


Figure C-59. Synchronization Points

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.

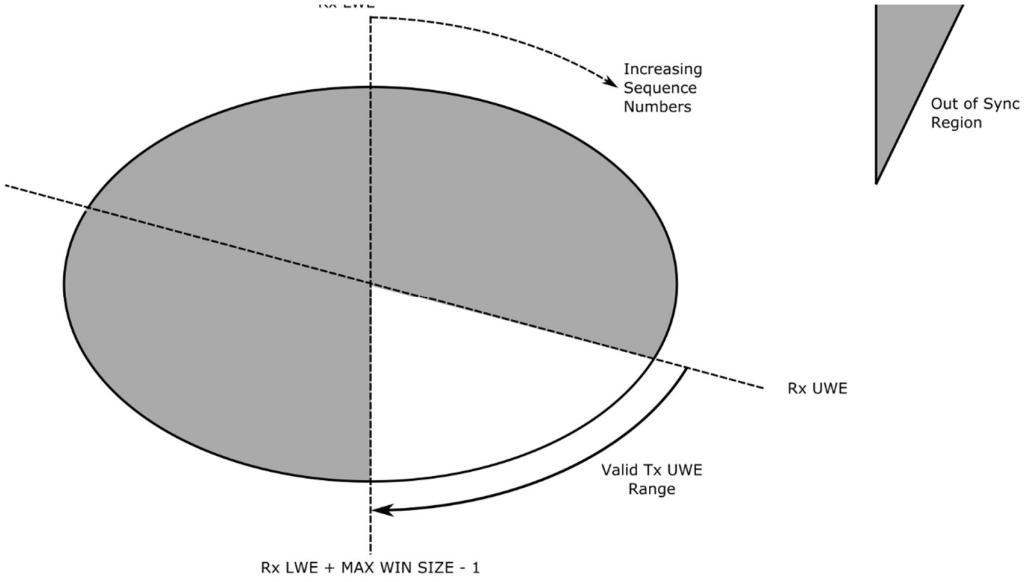


Figure C-60. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 1)

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.

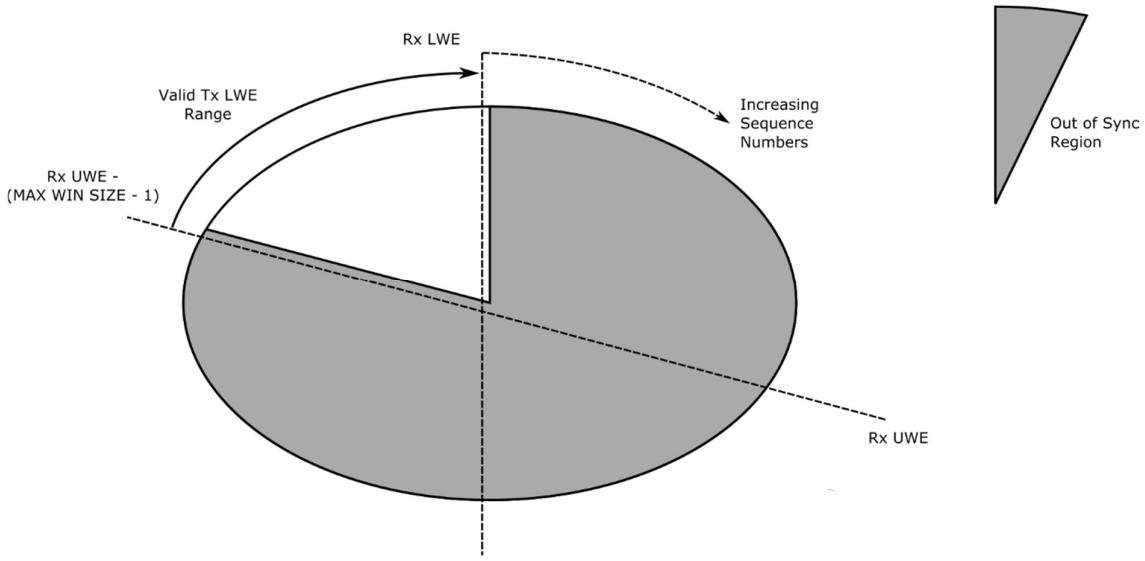


Figure C-61. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 2)

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.

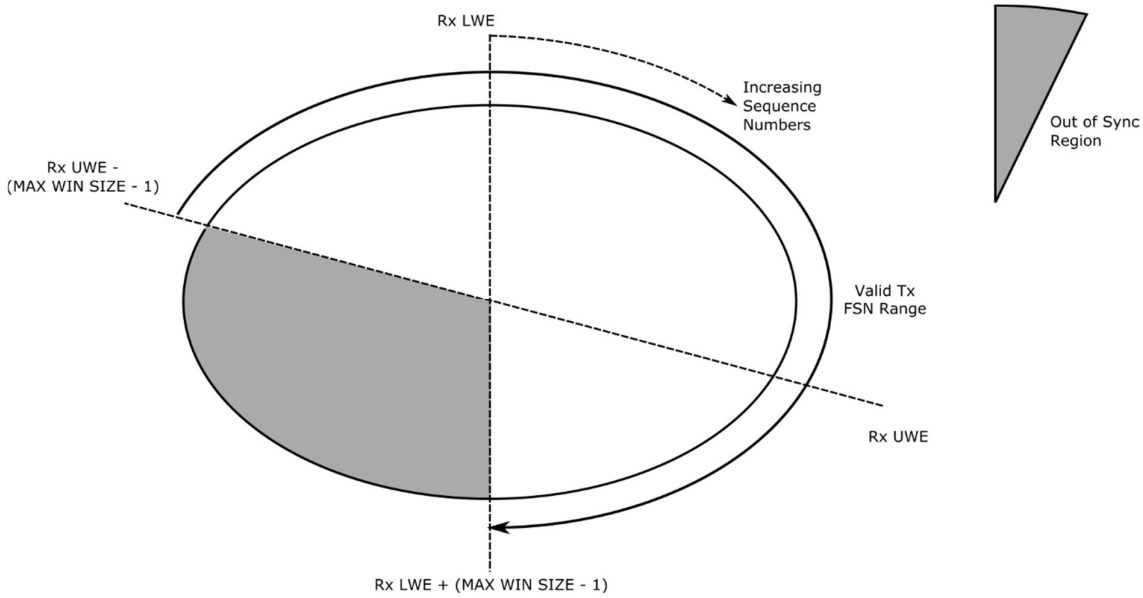


Figure C-62. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 3)

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.

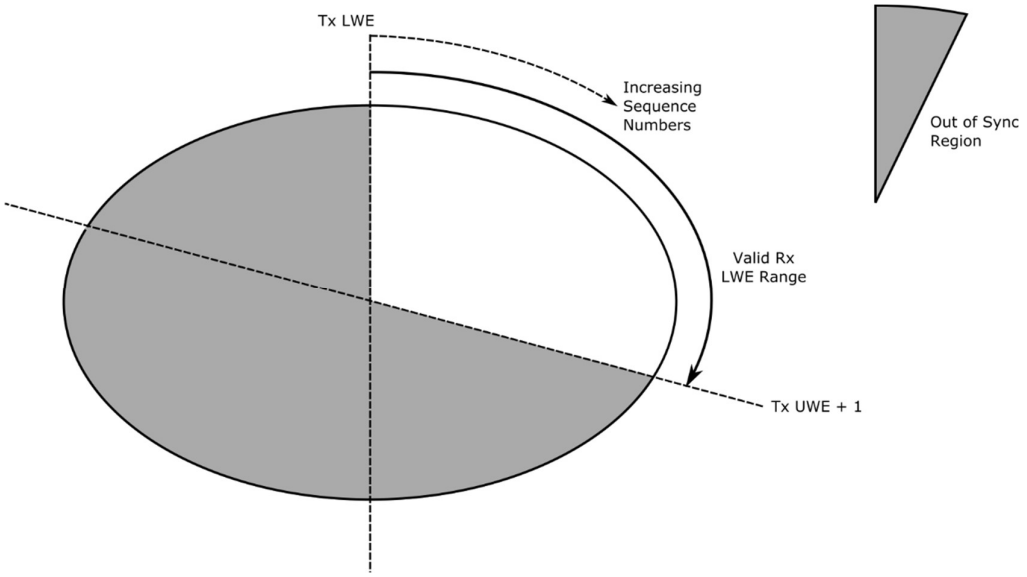


Figure C-63. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 4)

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.

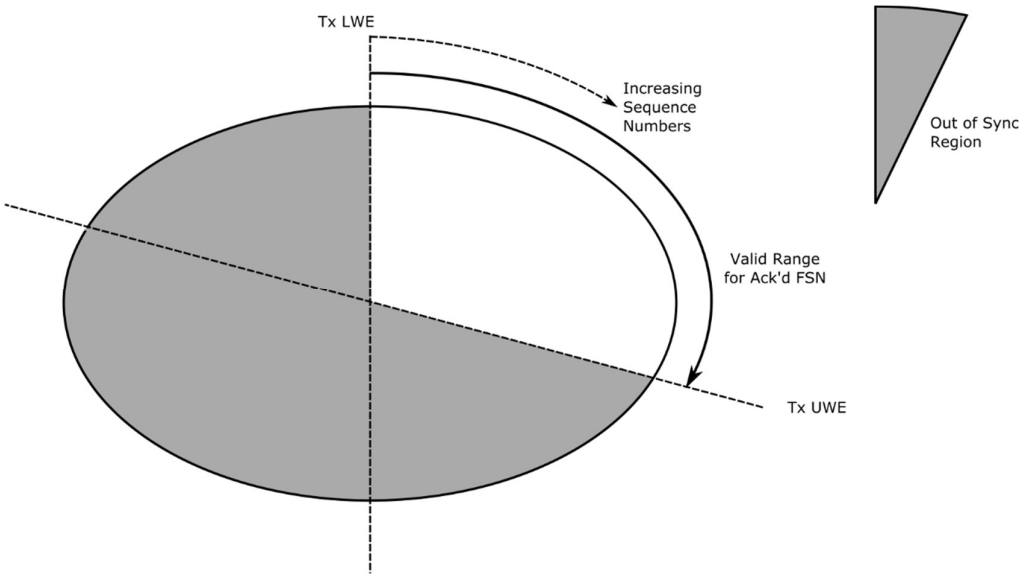


Figure C-64. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 5)

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).

ANNEX D INTERFACE BETWEEN DATA TRANSFER SUBLAYER AND COMMUNICATIONS EQUIPMENT (Mandatory)

The interface between the Data Transfer Sublayer and the communications equipment shall (1) be as defined in this Annex or as specified in Annex T (STANAG 5066 TRANSEC Crypto Sublayer using AES and other Protocols). This annex may be used to communicate from Data-Transfer-Sublayer to a modem or to a cryptographic device providing TRANSEC services.

To accommodate requirements imposed by the communications equipment, an arbitrary number of 'pre-fill characters may (1) be transmitted prior to the first valid DPDU in a transmission interval. Likewise, an arbitrary number of 'post-fill' characters may (2) be transmitted following the last valid DPDU in a transmission interval. The value of the EOT assigned to each DPDU in the transmission interval must be computed accounting for any post-fill characters in the transmission interval. Extraneous characters shall not be inserted between any valid DPDUs in the transmission interval.

Note that this interface defines a bit stream, and that a receiving implementation will need to analyse the stream looking for DPDU Maury-Styles headers to obtain byte alignment.

The interface specifies a synchronous serial digital data interface.

The line-drivers and receivers for the interface shall (4) be configurable for either balanced or unbalanced connection, in accordance with EIA-232D/423 for unbalanced connections and EIA-422 for balanced connections.

With respect to functional roles on the interface, the Data Transfer Sublayer shall (5) be hosted in a Data Terminal Equipment (DTE).

The clock source for the data output from the DTE (i.e, DTE data out) on the interface shall (6) be either configurable or from the DCE (i.e, either the cryptographic equipment or the modem).

The clock source for the data input to the DTE (i.e., DTE data input) shall (7) be from the DCE (i.e, either the cryptographic equipment or the modem).

The interface shall (8) provide full hardware-level handshaking for flow-control, in accordance with any standard recommendations.

Compatibility with MIL-STD-188-114 polarity, levels, and slew rates additionally may be required for interoperability with existing (cryptographic) equipment.

This Annex is broadly unchanged since Edition 1. Edition 4 has minor clarifications

and alignment to the new Annex T.

ANNEX E

INTENTIONALLY BLANK

ANNEX F SAP ASSIGNMENT (Mandatory)

Annex F defines SAPs that are assigned to applications using STANAG 5066 and default priorities for these applications.

F.1. Changes in This Edition

Most of Edition 3 Annex F has been split out into separate annexes to improve readability and maintainability. Details are set out in Section F.4.

F.2. SAP Assignment

Every application running over STANAG 5066 needs to use a SAP (range 0-15) and SAPs cannot be shared. Table F-1 specifies SAP assignment, with references to STANAG 5066 Annexes or to other open specifications.

Application	SAP ID	Reference
Subnet Management	0	Annex S (SIS Access Protocol)
Character-Oriented Serial Stream (COSS)	1	Annex P (ACP 127 & Character-Oriented Serial Stream)
ACP 142 (Military Messaging)	2	Annex Q (ACP 142)
HMTP	3	STANAG 5066 Edition 3 Annex F.5
HFPOP	4	STANAG 5066 Edition 3 Annex F.6
HF Operator Chat	5	Annex O (HF Operator Chat)
XMPP	6	XEP-0365 “Server to Server communication over STANAG 5066 ARQ” https://xmpp.org/extensions/xep-0365.html
ACP 142 (Email)	7	Annex Q (ACP 142)
Reserved	8	
IP Client	9	Annex U (IP Client)
Unallocated	10	
Unallocated	11	
Compressed File Transfer Protocol	12	Annex V (Compressed File Transfer Protocol)
Unallocated	13	
Unallocated	14	
Unallocated	15	

Table F-1: SAP Assignment

When protocols listed in Table F-1 are used, the assigned SAP ID **shall** be used. The Reserved SAP IDs wasused in Edition 3. It may be allocated or move to Unallocated in future editions of this annex.

ACP 142 is assigned two SAPs, as it is generally preferable to operate Email and

Formal Military Messaging on different SAPs.

When assigning additional applications to SAPs for a deployment, preference **shall** be given to Unallocated SAPs. Where this is not possible, assignment of another SAP not used in the deployment **may** be made.

F.3. Default Priorities

It is desirable that applications operate at consistent priorities for a deployment. For this reason, default priorities for the applications listed in Table F-1 are given in Table F-2. Deployments **may** choose to follow this recommendation or **may** make deployment-specific assignments.

Application	Default Priority
Character-Oriented Serial Stream (COSS)	5
ACP 142 (Military Messaging)	Based on Message Priority • OVERRIDE: 13 • FLASH: 11 • IMMEDIATE: 9 • PRIORITY: 7 • ROUTINE: 5 • DEFERRED: 3
HF Operator Chat	12
XMPP	10
ACP 142 (Email)	Based on Message Priority: • Urgent: 6 • Normal: 4 • Non-Urgent: 2
IP Client	6
Compressed File Transfer Protocol	4

Table F-2: Default Priority Assignments

Table F-2 sets out default priority assignments. ACP 142 (Military Messaging) has priorities derived from the military message priority. The defaults are assigned so that there is space above, below and in between each one. Other application assignments have been made relative to this.

F.4. Changes Since Edition 3

Most of Edition 3 Annex F has been moved to other Annex. Annex F retains its core function of specifying SAP assignment. The SIS Access Protocol, which was specified in Annexes A and F in Edition 3 is now specified in Annex S. Most of the other applications specified in Edition 3 Annex F have their own Annexes in Edition 4. This improves modularity and maintainability of STANAG 5066.

There are only 16 SAPs available, and it is anticipated that this will lead to requirements for careful allocation in some deployments. For this reason:

1. Edition 3 assignments that are believe to be un-used are removed.

2. Assignments **shall** be followed if an application is used, but **may** be re-assigned for other use.

The following protocols specified in Edition 3 do not have annexes in Edition 4, and do not have SAP assignments:

1. HMTP. Deployments have shifted to CFTP, which offers compression.
2. Ether Client. Only experimental implementations known, and future use unlikely.

BASIC FILE TRANSFER PROTOCOL (BFTP) and File-Receipt Acknowledgement Protocol (FRAP) are not specified in Edition 4.

UDOP and RCOP are not assigned SAPs, as the lack of protocol to support SAP sharing means that this approach has not been taken operationally.

Default Priorities have been added.

For the convenience of those familiar with Edition 3, the following table lists the disposition of each section of Annex F.

Edition 3 Section	Edition 4 disposition
F.1 (Standardized Client Requirements)	F.1 (SAP Assignment)
F.2 (Subnet Management Client/Server)	Dropped – nothing actually specified in Ed3
F.3 (CHARACTER-ORIENTED SERIAL STREAM (COSS) CLIENT	Annex P (ACP 127 & Character-Oriented Serial Stream)
F.4 (TACTICAL MILITARY MESSAGE HANDLING SYSTEM (T-MMHS) CLIENT - Interfacing STANAG-4406-Annex-E Compliant Systems w/ a STANAG 5066 Subnetwork)	Annex Q (ACP 142)
F.5 (HF MAIL TRANSFER PROTOCOL (HMTP): GUIDELINES FOR THE USE OF INTERNET SMTP OVER STANAG 5066 SUBNETWORKS)	Dropped as Annex, but Ed3 specification references from F.1
F.6 (POP3 FOR USE OVER STANAG 5066 TRANSPORT (HFPOP)	Dropped as Annex, but Ed3 specification references from F.1
F.7 (OPERATOR ORDERWIRE CLIENT (HFCHAT))	Annex O (HF Operator Chat)

Edition 3 Section	Edition 4 disposition
F.8 (RELIABLE CONNECTION-ORIENTED PROTOCOL)	Annex Q, Section 5
F.9 (UNRELIABLE DATAGRAM-ORIENTED PROTOCOL (UDOP)	Annex Q, Section 6
F.10 (EXTENDED CLIENT DEFINITION USING RCOP/UDOP APPLICATION IDENTIFIERS)	Relevant assignments now included in Annex Q (ACP 142) and Annex V (Compressed File Transfer Protocol). Other assignments not needed.in Edition 4. This section also includes specification of the BFTP and FRAP protocols. These are not included in Edition 4 as they are not widely used. The protocols specified in Edition 3 may be used with Edition 4 servers.
F.11 (ETHER CLIENT)	Ether Client is not included in Edition 4 as it is not widely used. Ether client as specified in Edition 3 may be used with Edition 4 servers.
F.12 (INTERNET PROTOCOL (IP) CLIENT)	Annex U (IP Client)
F.13 (RESERVED SERVICE ACCESS POINT IDENTIFIERS)	Addressed in Annex F Section 1.
F.14 (COMPRESSED FILE-TRANSFER PROTOCOL (CFTP) CLIENT	Annex V (Compressed File Transfer Protocol)
F.15 (UNASSIGNED SERVICE ACCESS POINT IDENTIFIERS)	Addressed in Annex F Section 1.
F.16 (RAW SIS SOCKET SERVER)	Included in Annex S (SIS Access Protocol)

ANNEX G

INTENTIONALLY BLANK

ANNEX H

INTENTIONALLY BLANK

ANNEX I

INTENTIONALLY BLANK

ANNEX J GENERAL REQUIREMENTS FOR ENHANCED MEDIA-ACCESS-CONTROL (MAC) CAPABILITIES (Informative)

The channel-access control capability of STANAG 5066 Annex B Edition 1 provides mechanisms (i.e., the CAS-1 linking protocol) to establish a point-to-point link (or links) for data communication. The CAS-1 protocol belongs to the class of link request/accept protocols that are effective at resolving the hidden-terminal problem in wireless point-to-point scenarios, by ensuring peer communication only.

This annex introduces modes for enhanced media-access control capability for HF data communication networks, and the prescribed method in which they are used with other STANAG 5066 capabilities. These channel-access modes extend or modify, but do not replace, the channel-access and link-control mechanisms defined in Annex B of this STANAG.

Enhanced Media-Access Control capabilities are defined in the context of an augmented model of the HF Subnetwork's protocol stack shown in Figure J-1.

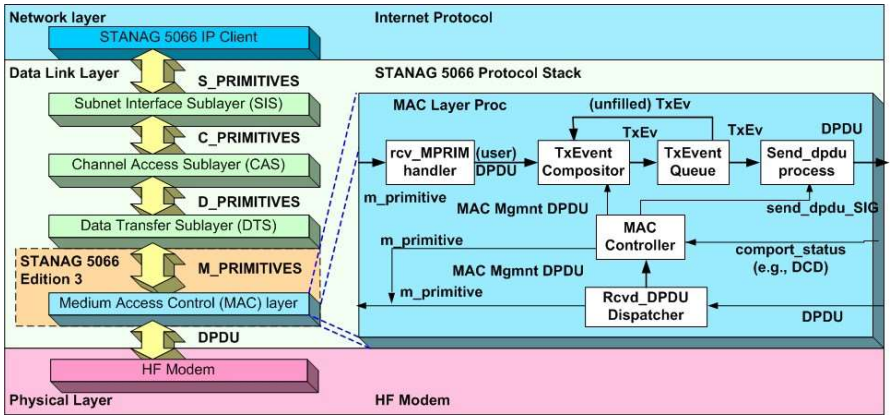


Figure J-1 — Augmented Model of the HF Subnetwork Protocol-Stack

The augmented model adds a Media-Access Control Sublayer (MACS) and inserts it below the Data-Transfer Sublayer of Annex C. Additional functionality implementing enhanced media-access control may be contained in the MACS, as shown in the figure.

The added media-access-control functionality:

- **shall** be based on the D_PDU message types defined in STANAG 5066 Annex C, and
- **may** use in addition new D_PDU types implemented for media-access control that **shall** conform to the message-definition rules and the Generic D_PDU Frame Structure and D_PDU field-element requirements of S'5066 Annex C Sections C.3.1 and C.3.2. New functionality to implement enhanced media-access-control **shall** be confined to the D_PDU Type-Specific Header element and, if present, D_PDU Payload element of any new D_PDU type.

To minimize impact on pre-existing functionality defined in Annexes A, B, and C for other layers of the HF subnetwork, new D_PDU types defined for media-access control **shall** only support peer-to-peer MACS-layer communication between nodes.

Node-to-node communication between HF subnet clients and peer-layer-to-peer-layer communication supporting the functionality of Annexes A, B, and C **shall** continue to use the D_PDU types defined in Annex C. The S_PDU and C_PDU specifications of Annex A and Annex B **shall** be unchanged by the MACs functionality.

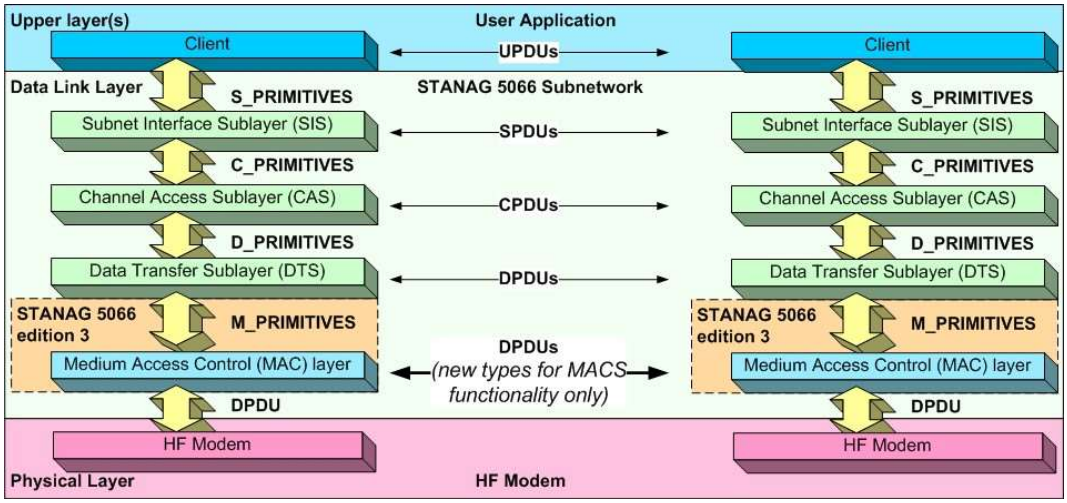


Figure J-2 — Peer-to-Peer Communication for Enhanced Media-Access Control

MACS functionality may be tailored as outlined in section J.3, which defines options to implement a range of functionality for multi-node networks. Detailed functional and performance specifications from other Annexes are cross-referenced where appropriate.

Four Media-Access Control Modes are defined:

1. Point-to-Point Mode (P2P)
2. Carrier-Sense Multiple Access (CSMA) Media Access
3. Wireless Token-Ring Protocol (WTRP) Media Access
4. Adaptive Time-Division Media Access (ATDMA)

These modes are summarized below. Implementation and performance requirements for these are standardized in the cross-referenced annexes.

J.3.1 Point-to-Point (P2P) Media-Access Mode

The point-to-point (P2P) media-access mode enables two nodes to reserve the channel for their use in point-to-point communication. The default point-to-point channel- access control **shall** be implemented in accordance with STANAG 5066 Annex B. It **shall** also use CSMA following STANAG 5066 Annex K, noting that for two nodes only some simplification of that Annex K procedure is possible.

J.3.2 Carrier-Sense Multiple Access (CSMA) Media-Access Mode

The Carrier-Sense Media Access (CSMA) mode enables a set of nodes to share the channel in a multi-node network. CSMA is a form of media-access control based on a node's ability to listen to the channel (i.e., radio-frequency carrier) and use its local knowledge that the channel is clear to control its transmissions to avoid interference — also called collisions — with the transmissions of other nodes.

If a CSMA mode is implemented, it **shall** be implemented in accordance with STANAG 5066 Annex K.

The CSMA mode specified in Annex K conforms to the requirements of this Annex: it is implemented using only the message catalogue defined by STANAG 5066 Edition 1, and the basic capabilities of Annexes A, B, and C, with augmented requirements for Annex D to implement a Listen- Before-Transmit (LBT) mechanism with collision avoidance.

J.3.3 Wireless Token-Ring Protocol (WTRP) Media-Access Mode

The Wireless Token-Ring Protocol (WTRP) media-access mode enables a set of nodes to share the channel in a multi-node network. Token-ring or token-bus multiple access is a form of media-access control based on ownership of a 'token' that grants the right to transmit (RTT) on the channel. A token holder transmits until it no longer has data to send, or until its right-to- transmit timer expires, and then it passes the RTT

token to its successor. Adaptation of each node's allocated channel capacity to the offered traffic load occurs automatically with the WTR mode.

The Wireless Token-Ring Media-Access mode **shall** be implemented in accordance with STANAG 5066 Annex L. The Wireless Token-Ring Protocol (WTRP) defined in that Annex is in two parts:

- a D_PDU message design for the management tokens exchanged by network nodes, and
- the algorithms used to create, maintain, and repair the ring (i.e., transmission sequence) of nodes in the network.

WTRP's D_PDU message design conforms to the requirements of this Annex. It uses new D_PDU message types based on EXTENSION D_PDU that conforms fully to the requirements of Annex C, whose D_PDU- specific part provides the requisite data fields to satisfy the information exchange requirements of WTRP.

J.3.4 Adaptive Time-Division Media-Access (ATDMA) Mode

The Time-Division Media-Access (TDMA) mode enables a set of nodes to share the channel in a multi-node network. TDMA divides the channel access into timeslots that are allocated to nodes in the network (or, even more generally, allocated to network services).

Fixed TDMA protocols are known to be inefficient in channel utilization and service times when the traffic offered by the timeslot owner (i.e., the node or service to which the time slot is allocated) is mismatched to the channel capacity provided by the time slot. This is particularly true when the node or service has insufficient offered traffic to fill the timeslot(s) it has been allocated.

Adaptive TDMA (ATDMA) permits variations in the timeslot allocation or length that allow the network to adapt to variations in the traffic offered by nodes and in their service requirements. To use ATDMA's adaptivity with STANAG 5066 would require new D_PDU management message types with which nodes coordinate timeslot usage and length information.

There is currently no STANAG 5066 Annex that specifies TDMA or ATDMA.

Through their conformance with the generic D_PDU-message structure defined in Annex C the enhanced media access protocols defined and cross-referenced herein are compatible in the limited sense that their common message elements (e.g., Maury-

Styles synchronization preamble, D_PDU type field, address fields, etc.) are recognizable regardless of which MACS mode is in operation or has been implemented by the node. Thus, all STANAG-5066-compliant implementations will be capable of decoding the field elements common to all the D_PDU messages. And, in particular, all compliant implementations will be capable of determining the source address of the node that sent the D_PDU and the type number of the D_PDU that was sent, even if the node does not implement the MACs protocol for which the D_PDU is used.

But the enhanced media access modes defined herein are not interoperable. There is no intent or expectation that a node implementing one of these enhanced media-access modes should be interoperable with a node implementing a different protocol. Rather, the intent and expectation is that standard operating procedures for network establishment will ensure that only nodes using the same media-access control protocol are network members. As this is a naïve view of what can happen in an operational network, this Annex defines provisions for nodes to discover the MAC-mode in use by other nodes, to, at the very least, recognize when they are using incompatible protocols and to take appropriate action to avoid mutual interference.

J.4.1 MAC-Mode Discovery

The processes by which nodes discover the media-access-control modes in use within a STANAG 5066 network are called MAC-Mode Discovery.

Nodes **should** implement MAC-Mode Discovery, which consists of the process elements defined here:

- Use of D_PDU Type 15 Warning Messages – the (mandatory) provisions of Annex C Section C.3.12 **shall** apply to the implementation of MAC-Mode Discovery, as further amplified below.
- Channel Analysis – nodes implementing a given media-access mode shall analyze the channel usage by other nodes to detect violations of the protocol that it implements, and take actions as noted further below. These actions include the use of Type 15 Warning messages to notify neighbouring nodes of the protocol incompatibility.

J.4.1.1 Use of Type 15 Warning Messages

A STANAG 5066-compliant node that receives a D_PDU message that is “unexpected or unknown” to it is required under the provisions of Annex C Section C.3.12 to send a Type 15 D_PDU (Warning) Message to the node that generated the unexpected or unknown D_PDU. This requirement is unchanged by the enhanced-media-access modes defined herein.

In particular, and with cross reference to Annex C Section C.3.12, a node that receives a D_PDU that is unexpected for the MACS protocol is it is using (this may be an indication that the sending node is using a different MACS protocol) or that unknown to it (also a potential indication of an incompatible MACS protocol) will determine the source address of the node that sent the unexpected or unknown D_PDU and send a Type 15 D_PDU (Warning) Message to that node:

- As required in Annex C, the Type 15 D_PDU (Warning) Message RECEIVED FRAME TYPE field **shall** indicate the Type number of the unexpected or unknown D_PDU.
- If the node does not recognize the received D_PDU, the Type 15 D_PDU (Warning) Message REASON WARNING SENT field **shall** be set to the value assigned to the reason “Unrecognised D_PDU type Received”.
    - N.B.: This case applies to D_PDU types with subtypes, i.e., the Type 6 D_PDU (Management) that uses the Engineering Orderwire (EOW) Type field (as defined in Annex C) as a subtype indicator to distinguish variants of the Extended Management Message types: the receiving node may recognize the main type (i.e., Type 6 Management Message) but not the subtype (i.e., the particular Extended Management Message subtype designated by the EOW-type field type). In this case, the node **shall** treat the D_PDU as an unrecognized D_PDU, and send a Type 15 D_PDU (Warning) Message as above.
- If the node recognizes the received D_PDU, but it was unexpected for the protocol that it was using, the Type 15 D_PDU (Warning) Message REASON WARNING SENT field **shall** be set to the value assigned to the reason “Invalid D_PDU Received for Current State”.
    - N.B.: This case also applies to D_PDU types with subtypes as described in the preceding case. If the node receives a D_PDU of the proper type but if the subtype is unexpected, e.g., it designates an Extended Management Message defined for a MACS protocol that the node recognizes but is not currently executing, then in this case also, the node **shall** treat the D_PDU as an unexpected D_PDU, and send a Type 15 D_PDU (Warning) Message as above.

- In particular, a node executing the P2P or CSMA mode that receives any of the new D_PDU types defined in Annex L (for WTRP) (whether or not they are recognized to the receiving node) **shall** send a Type 15 D_PDU as specified

above.

Use of Type 15 D_PDU (Warning) Messages of course is possible only when the node is configured in a compatible transmission mode; nodes configured for transmit-only operation (e.g., to providing exclusive support to a Broadcast Data Exchange Session as defined in Annex A) will not receive any D_PDUs that could trigger the warning condition, and nodes configured for receive-only operation (e.g., to receive the Broadcast Data) would not be capable of sending the warning messages and therefore disable the protocol-error-detection logic and warning-message- generation functionality.

J.4.1.2 Channel Analysis

Transmissions by nodes executing one MAC-mode will likely be uncoordinated with the transmissions of nodes executing another (i.e., they don't participate in the protocol), and mutual interference may occur.

Nodes executing a given MAC-mode **shall** analyze the D_PDU messages received from other nodes to detect violations of the protocol it is executing. Message analysis includes the following:

- processing of received Type 15 D_PDU (Warning) Messages in accordance with Annex C and as amplified by Section J.2.1.1.
- analysis and identification of the MAC mode in use by other nodes:
    - transmissions that contain none of the new D_PDU types (recognizable to the receiving node) defined in Annex L (for WTRP) **should** be assumed to originate from a node that is currently operating in P2P or CSMA modes, as these modes (i.e., P2P and CSMA) do not use any new D_PDUs for their operation. This indication is one of the current operating mode, and not of capability.
    - Transmissions that contain Type 15 D_PDU (Warning) Messages that warn ignorance of any of the new D_PDU types defined in Annex L (for WTRP), i.e., that have a RECEIVED FRAME TYPE field defined for one of the new D_PDU types and that denotes a REASON WARNING SENT field as unrecognized, **should** be assumed to originate from a node that is capable of operating only in P2P or CSMA modes. The presumption **should** be that node is a legacy (STANAG 5066 Edition 1 compliant) implementation incapable of executing either WTRMA or ATDMA.

- Transmissions that contain Type 15 D_PDU (Warning) Messages that warn of unexpected use of any of the new D_PDU types defined in Annex L (for WTRP), i.e., that have a RECEIVED FRAME TYPE field defined for one of the new D_PDU types and that denotes a REASON WARNING SENT field as unrecognized, **should** be assumed to originate from a node that is operating in a different MACS mode than the node to which the Type 15 D_PDU (Warning) Message is addressed.

J.4.2 Embedding Broadcast / Multicast Traffic

Receive-only nodes (e.g., nodes in an emission-control [EMCOM] status) make no transmissions that can interfere with the operation of multi-node network, whatever MACS mode it uses. This may be exploited by nodes that do transmit in a multi-node network to embed broadcast or multicast traffic, a capability that is fully consistent with the intent of Annex A Section A.1.1 (from Edition 2 and later of this STANAG).

The MAC layer **may** be mapped onto the underlying HF channel in one of the three ways set out below.

J.5.1 Fixed Frequency

In this mode, a fixed frequency agreed by all nodes is used. If STANAG 5069 Wideband HF is used, a fixed bandwidth is also chosen. This **shall** be the same for all nodes on the network. This is a simple and robust choice that is effective for some deployments, such as a network using HF surface wave.

J.5.2 Scheduled Frequency

A variant on fixed frequency is to have an agreed schedule of frequency use that is followed by all nodes. This can allow for adaptation to conditions in a simple manner, for example to use different daytime and nighttime frequencies.

J.5.3 ALE for All Nodes

STANAG 5066 Annex B defines channel access mechanisms that set up ALE links on demand for point to point and multicast links.

An alternate approach is to use ALE for all nodes on the channel, which is managed at MAC layer. By maintaining an open channel, it enables all nodes on the channel to

communicate with CSMA or WTRP. It will prevent the situation where a pair of nodes are connected on a channel, which blocks communication with other nodes.

Use of ALE at MAC level means that a channel can be kept open for all nodes and that the best channel at a given time can be chosen. It is important that the ALE channel is closed and re-opened from time to time, in order to ensure choice of best channel and to bring on nodes that were not available on the previous ALE setup.

1. Changed to include networks of any number of nodes. Edition 3 title and introduction suggest multiple nodes, but this is contradicted by much of the text which covers two node networks.
2. Removed normative references to Annex M, which is a placeholder.
3. Require use of CSMA for point to point. Edition 3 notes that MACS adds no functionality which is contradicted by footnote explaining why MACS should be used.
4. Added section on mapping to channel and use of ALE.

ANNEX K HIGH-FREQUENCY CARRIER-SENSE MULTIPLE-ACCESS PROTOCOL (Optional)

K.1 INTRODUCTION

This Annex specifies a High-Frequency Carrier-Sense Multiple Access (CSMA) with Collision Avoidance protocol [1] for STANAG 5066 in multi-node single-frequency networks.

The HF CSMA protocol introduces no new DPDU types to the STANAG 5066 catalogue. There is no explicit peer-to-peer communication required by the CSMA protocol. CSMA media access control relies on a node's local information of HF channel activity and inferences of the activity — or lack thereof — of other nodes.

In order to be effective, the options used in this annex and the values for timers need to be set consistently for all nodes on the network.

Annex K is Optional. However if CSMA or any other form of Listen Before Transmit is implemented with STANAG 5066, it **shall** conform to Annex K.

Section K.2 of this Annex presents an overview of the protocol and provides a definition of terms. Details of the protocol, its state diagram, and parameter values are specified in section K.3

K.1.1 Changes in This Edition

The functional differences between this specification and Edition 3 are set out in Section K.5.

The key change relative to Edition 3 is to add some new timer values, that enable better performance on a network with a small number of nodes.

K.2 OVERVIEW: CARRIER-SENSE MULTIPLE-ACCESS PROTOCOLS

Definitions and management concepts are introduced below prior to detailed specification of the protocol in later sections.

K.2.1 Background

The protocol specified in Edition 3 was based on a presentation to the High-Frequency Industry Association in 2002 by Robert McFarland of Rockwell Collins., "Collision Avoidance using STANAG 5066 in a Network Environment", with additional input from Michael Stringer of Harris Corporation. This used a Jitter mechanism so that many nodes on a network could reduce the chance of collision. Standardizing this enables multi-vendor deployments.

The update for Edition 4 uses work from Isode specified in the paper "Slotted Option for STANAG 5066 Annex K". This introduced the following improvements:

1. An optional "slotted" mechanism where each node is assigned a slot. This gives faster switching and higher resilience for small networks.

   a. The “jitter” mechanism remains an option for larger networks, although modern large networks will generally use ALE rather than fixed frequency.
2. Optimized switching when just two nodes are communicating. This removes any residual benefits of operating without CSMA (Annex K) or WTRP (Annex L).
3. Use of different timings when STANAG 5066 EOT is detected to enable faster switching than is possible with a single timer.
4. Optimizing repeat transmissions from a single node, which is important for non-ARQ bulk protocols such as ACP 142.

This combination leads to Annex K providing an alternative to WTRP (Annex L) which will generally provide better performance for a lightly loaded network.

K.2.2 Definitions

The following terms are used in the specification of the High-Frequency Carrier-Sense Multiple Access (CSMA) protocol.

K.2.2.1 Stations and Nodes

The terms “*station*” and “*node*” are used interchangeably to describe the communication entities on the shared HF channel.

K.2.2.2 Collisions

Channel collisions — or, simply, *collisions* — are simultaneous or overlapping transmissions by two or more nodes that interfere with each other, preventing reception by another node.

K.2.2.3 Listen-Before-Transmit (LBT)

Listen-before-transmit is the action a node takes to ensure that the channel is unoccupied and free of activity before it attempts to transmit.

K.2.2.4 DCD

DCD is the *Data-Carrier-Detect* signal provided by the communications equipment interface. A node senses that an HF radio carrier is present when DCD is *true* (or — using the nomenclature of some interfaces — asserted). Communications equipment that senses an idle channel will provide set DCD = *false* (noted in this annex as the !DCD signal). DCD does not always give a clear indication of when a transmission ends, due to channel fades. Note that DCD requires an interface from modem to STANAG 5066, which might need a crypto bypass.

K.2.2.5 VDCD

VDCD is a *Virtual Data-Carrier-Detect* signal, derived from reception of valid DPDUs. Observation of DPDU header's End-of-Transmission (EOT) field (defined in Annex C Section C.3.2.3) **shall** be used to extrapolate into the future and predict the time at which a channel will be idle. The EOT mechanism provides a robust mechanism to determine end of transmission. It is preferred to DCD, and when VDCD is available it enables faster switching.

K.2.2.6 Contention Interval

The *contention interval* is the period of time during which nodes attempt to access the shared HF channel.

K.2.2.7 Collision Avoidance

Collision Avoidance is a strategy that nodes use during the contention interval to increase the probability that some contending node successfully accesses the channel.

K.2.2.8 Node States

A node may be in one of the following states as it executes the CSMA protocol:

- *Offline* state (OFFLINE) — the *offline* state is a state in which a station acts as if it were physically offline, i.e., it can neither transmit nor receive;
- *Sensing* State (SENSE) — the *sensing* state is a state in which a station monitors channel activity (using the DCD or VDCD signals), waiting for it to become idle.
- *Listen-before-Transmit-Wait* State (LBT_WAIT) — the *Listen-before-Transmit-Wait* state is a state in which a station waits to determine if another node has transmitted on the idle channel.
- *Contention-Wait* State (CONT_WAIT) — the *contention-wait* state is a state in which the station waits a random time before it may transmit. Nodes wait a random time during this state while continuing to sense channel activity to avoid collisions.
- *Linking* State (LINKING) — the *Linking* state is a state in which a station can transmit data.

K.2.2.9 Timers

The following timers control operation of the CSMA protocol:

- LBT_WAIT_TIMER – the LBT_WAIT_TIMER controls the maximum length

of time a node will wait on an idle channel before transitioning to the contention-wait state. This value is determined from a number of parameters.

- CONT_WAIT_TIMER – the CONT_WAIT_TIMER controls the length of time a node (i.e., a contending node) waits to access the channel before it starts to transmit.

K.2.2.10 Scalar Control Parameters

The following scalar parameters control operation of the CSMA protocol:

- LBT_WAIT_TIMER_VALUE – the waiting time set for the LBT_WAIT_TIMER determined from a number of parameters;
- LBT_WAIT_TIMER_VALUE_DCD – the waiting time set for the LBT_WAIT_TIMER when end of transmission is determined by DCD;
- LBT_WAIT_TIMER_VALUE_VDCD – the waiting time set for the LBT_WAIT_TIMER_GC when end of transmission is determined by VDCD;
- CONT_WAIT_TIMER_VALUE – the waiting time set for the CONT_WAIT_TIMER; this is a computed value that depends on other parameters and whether Jitter or Slotted is used.;
- NUM_CONT_SLOTS – the number of slots defined. For slotted, this **shall** be greater than or equal to the number of nodes on the network;
- CONT_SLOT_WIDTH – the duration of each contention slot.
- NODE_SLOT_POSITION: For slotted, a per node configuration in the range 1 to NUM_CONT_SLOTS. Each node **shall** have a different value configured.

K.2.3 Concept of Operations

A likely occurrence in multi-node HF subnetworks is that nodes need to communicate at the same time and may attempt to do so. Uncontrolled attempts to access and transmit on the same channel can lead to channel collisions when these transmissions occur at the same time. Mitigating the effects of collisions requires retransmissions and delays that lower channel throughput, but, when retransmissions remain uncontrolled, can further decrease throughput. Repeated collisions on an uncontrolled channel can degrade the network severely. Carrier-Sense Multiple Access (CSMA) concepts that include collision-avoidance (CA) can provide simple yet effective mechanisms for a (small) number of nodes to share a single- frequency HF subnetwork when they each have traffic to send.

The concept of operations for the CSMA protocol follows.

From an offline, non-operating state a node starts the protocol by entering a carrier-sensing state, where it listens on the channel for a radio-frequency carrier. The mechanisms a node uses for carrier-sense may vary but this Annex assumes that the carrier is sensed using either the Data-Carrier-Detect (DCD) signal available from the communications equipment, or from a Virtual Data-Carrier-Detect signal generated by tracking the EOT-field of received DPDUs contained in previous transmissions.

A node with no data queued for transmission remains in the carrier-sensing state.

A node with queued data to send that senses an idle channel enters the Listen- Before-Transmit state, and sets a timer — the LBT_WAIT_TIMER — that controls its exit from the state. The timer is restarted whenever the carrier is detected. If the LBT_WAIT_TIMER expires and the node still has queued data, the node enters a contention state. At any time in the Listen-Before-Transmit state, if the node no longer has any queued data, the node returns to the carrier-sensing state. Note that the LBT_WAIT_TIMER timer value specifies the maximum time a node will wait on an idle channel. The node will wait for a longer time — potentially a much longer time — on a busy channel because the LBT_WAIT_TIMER is restarted whenever carrier is sensed.

Nodes wait before contending for the channel and transmitting because of the time delays associated with the carrier-sense mechanism. A node waits to determine that the channel truly is idle. Detection of a busy channel — i.e., declaration of the DCD or VDCD signals — is not instantaneous. A node's transmissions will have a propagation delay between itself and any receiver. Then, DCD is declared by a receive modem on declaration of valid data and it takes the modem's decoder some number of interleaver frames to make this declaration, introducing additional delay. Additionally, for secure systems that use crypto between the HF modem and the STANAG 5066 system, the modem DCD either needs to use a crypto bypass or to be relayed through the cryptographic equipment, which may introduces additional delay. These delays influence the length of the listen- before-transmit interval that enables nodes to avoid — but not completely prevent — collisions on the channel.

On completion of the Listen-before-Transmit interval i.e., when the LBT_WAIT_TIMER expires, a node with queued data to transmit enters a contention state, where it uses one either a jitter or a slotted mechanisms set out below.

For both mechanisms, the contention-state's random waiting interval is quantized. It consists of a number of slots, each of a fixed width. The slot width is specified to insure that all nodes in the network can detect a node's transmission before the beginning of the next slot.

The slot width **shall** be chosen so that it is greater than the channel-sensing delays — i.e., the sum of the modem-, crypto-, propagation- and other delays.

K.2.4 Jitter Mechanism

With the jitter mechanism, a node selects a random interval to wait before it enters the transmit state. A node picks a random time to wait by picking a random slot to start its

transmission.

The random waiting period during the contention interval is intended to avoid collisions — the carrier-sense and listen-before-transmit states tend to synchronize the access attempts of multiple nodes and the random waiting time during the contention interval forces, with some probability of success, the attempts to occur at different times. With good choice of slot width, the probability of collision can be reduced to the probability that two or more contending nodes pick the same contention slot.

Quantizing the contention-state's waiting time and constraining nodes to attempt transmissions only at slot boundaries will reduce average wait time by reducing collisions. Unconstrained transmission- attempt times during the contention interval increases the possibility of collisions between contending nodes.

K.2.5 Slotted Mechanism

With the slotted mechanism, there is a slot assigned for each node on the network, and the total number of slots is equal to the number of nodes on the network.

Advantages of the slotted mechanism are:

1. Each node will transmit in its own slot, so there is no risk of nodes picking the same slot.
2. For small networks it is safe to use a small number of slots.
    a. Note that for a two-node network, slots are not needed at all, because of a mechanism described later:

Disadvantages of the slotted mechanism are:

1. It requires more coordinated configuration than is needed by the jitter mechanism.
2. It is inefficient for larger networks, where Jitter is preferable.
3. It is not "fair" as priority is given to nodes with earlier slots. This may be an advantage in some deployments.
    a. The unfairness only becomes noticeable under high load. For a highly loaded channel, WTRP (Annex L) is likely to be preferable.

K.2.6 Optimizing for Two Active Nodes

Use of CSMA naturally introduces delays between transmissions in order to avoid collisions, while allowing multiple nodes to share a network. A common scenario on a lightly loaded network, for which Annex K is targeted, is that active communication is restricted to a pair of nodes. Annex K optimizes for this scenario, by allowing a node to transmit at the earliest safe point, when it knows that there is only a pair of nodes communicating. This gives useful performance optimization for this common scenario.

Note that either active node can introduce communication with a third node at any time, by establishing a CAS-1 link. However, this mechanism does block communication between other nodes until the pair of nodes has finished communicating. This is not

expected to be a problem on target networks (lightly loaded).

K.3 SPECIFICATION AND PROTOCOL

The CSMA protocol is specified below in the following sections:

- K.3.1 Overall State Diagram , which presents the protocol as a state graph with directed transitions and transition events that may be conditional or unconditional;
- K.3.2 State Specifications, which presents detail of each state and outbound-transition tables that govern the actions that take place when the node is in the state, and the events that trigger transitions to the next state.

K.3.1 Overall State Diagram

Figure K-1 below shows the simplified CSMA state diagram in which only the states and transition events are shown. Each state is specified in detail in the following section.

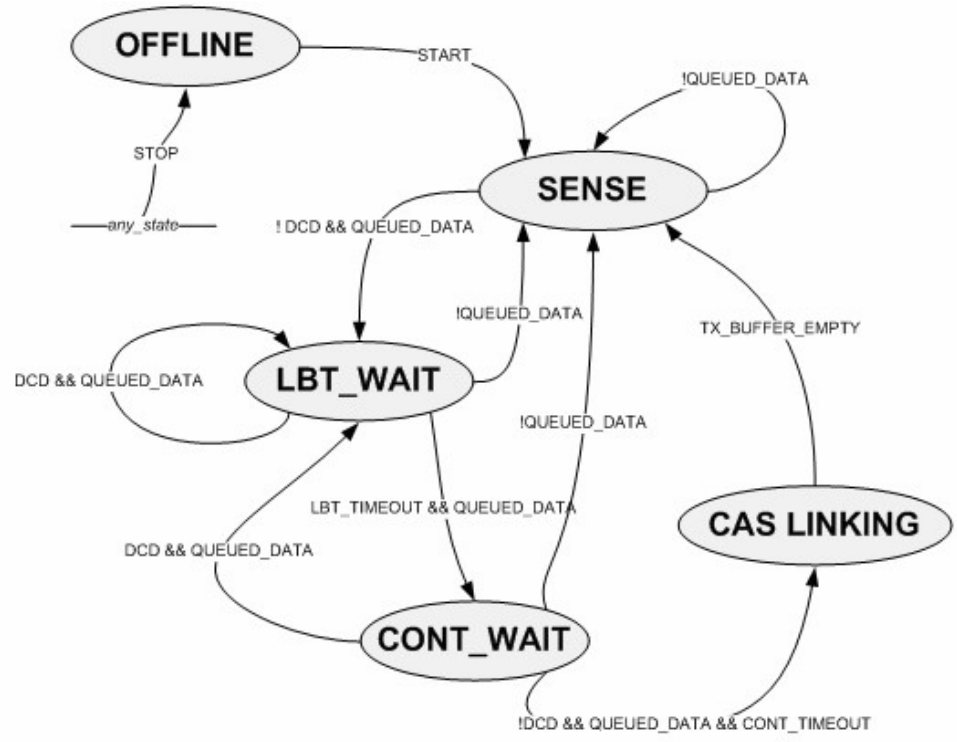


Figure K-1 — CSMA state diagram.

K.3.2 State Specifications

Specifications for CSMA operation for each state are provided below. For specification-

purposes, transitions from each state to the next state shall be controlled by the accompanying outbound-transition tables, which specify:

- The current state,
- The event that triggers the transition,
- The action that shall be taken as a result of the transition,
- The next state to which the protocol transitions,
- The timer (if any) that is started.

K.3.2.1 Offline State (OFFLINE)

A node in the OFFLINE state **shall** neither send nor receive data.

Outbound transitions from the OFFLINE state **shall** conform to the table below.

Table 1 — OFFLINE-State Outbound-Transition Table

state	event	condition	action	next state	start timer
OFFLINE	START	start event received from the subnetwork	Neither send nor receive	SENSE	*none*

A node **shall** transition to the SENSE state when it receives a START signal from the subnetwork management function (i.e.., when the nodes starts operating).

K.3.2.2 Carrier-Sensing State (SENSE)

A node in the SENSE state **shall** listen for an idle channel, which is declared whenever the DCD signal is false (i.e, a !DCD signal).

Outbound transitions from the SENSE state **shall** conform to the table below.

Table 2 — SENSE-State Outbound-Transition Table

state	event	condition	action	next state	start timer
SENSE	!DCD (idle channel)	data QUEUED for transmission		LBT_WAIT	LBT_WAIT_TIMER
SENSE		no data for transmission		SENSE	*none*

A node with no data queued for transmission **shall** remain in the SENSE state

regardless of channel activity.

A node with data to send **shall** wait in the SENSE state until it detects an idle channel, and then **shall** transition to the LBT_WAIT state. The value that LBT_WAIT_TIMER is to set depends on a number of conditions.

1. LBT_WAIT_TIMER **shall** be set to zero if the node determines that the only active CAS-1 link is between the node and one other node. This is determined if one the following conditions are true:
    a. If the received transmission contains D_PDUs directly addressed to the local node and there are no D_PDUs directly addressed to other nodes. Note that the received transmission **may** contain non-ARQ D_PDUs addressed to broadcast and/or multicast destinations; or
    b. A transmission is received, but the local node is not able to parse any D_PDUs in the transmission, and the local node is known to be a peer in all known CAS-1 transmissions. Under these conditions it is able to transmit safely, as no other node is expected to transmit at this point. In order to determine this, a node needs to monitor the status of all CAS-1 links on the channel (not just the CAS-1 links that the node is involved in).

The first condition **shall** be checked. The second condition **may** be checked.

2. LBT_WAIT_TIMER **shall** be set to LBT_WAIT_TIMER_VALUE_DCD if the !DCD event was determined by DCD and not by VDCD.
3. LBT_WAIT_TIMER **shall** be set to LBT_WATI_TIMER_VALUE_VDCD if !DCD event was determined by VDCD (EOT).

K.3.2.3 Listen-Before-Transmit-Wait State (LBT_WAIT)

A node in the LBT_WAIT state **shall** wait while continuing to sense channel activity, to ensure that the channel is truly idle.

Outbound transitions from the LBT_WAIT state **shall** conform to the table below.

Table 3 — LBT_WAIT-State Outbound-Transition Table

state	event	condition	action	next state	start timer
LBT_WAIT	DCD (busy channel)	data QUEUED for transmit		LBT_WAIT	LBT_WAIT_TIMER (restarts timer)
LBT_WAIT	LBT_WAIT_TIMER timeout event	data QUEUED for transmission	compute CONT_WAIT_TIMER_VALUE	CONT_WAIT	CONT_WAIT_TIMER
LBT_WAIT		no data for transmission		SENSE	*none*

At any time while in the LBT_WAIT state, detection of a busy channel (i.e., receipt of a DCD or VDCD signal) **shall** restart the LBT_WAIT_TIMER with a value of LBT_WAIT_TIMER_VALUE_DCD; the node **shall** remain in the LBT_WAIT state in this case.

At any time while in the LBT_WAIT state, detection of an empty data queue (e.g., it may have been emptied by a management function) **shall** force a transition to the SENSE state.

If the LBT_WAIT_TIMER expires, CONT_WAIT_TIMER is set and calculated in the following manner.

1. If the Jitter approach is taken, CONT_WAIT_TIMER **shall** be set to multiple of CONT_SLOT_WIDTH, multiplied by a random number in the range 0 to NUM_CONT_SLOTS-1.
2. If the slotted approach is taken and the last transmission was not made by the local node, CONT_WAIT_TIMER shall be set to:

(NODE_SLOT_POSITION -1) * CONT_SLOT_WIDTH

3. If the last transmission was made by the local node, CONT_WAIT_TIMER shall be set to:

NODE_SLOT_POSITION * CONT_SLOT_WIDTH

The third condition is to address sending Non-ARQ data. When sending ARQ data, it is expected that another node will transmit next. When sending Non-ARQ, this may not be the case.

K.3.2.4 Contention-Wait State (CONT_WAIT)

Outbound transitions from the CONT_WAIT state **shall** conform to the table below.

Table 4 — CONT_WAIT-State Outbound-Transition Table

state	event	condition	action	next state	start timer
CONT_WAIT	DCD (busy channel)	data QUEUED for transmission		LBT_WAIT	LBT_WAIT_TIMER (restarts timer)
CONT_WAIT	!DCD && CONT_WAIT_TIMER	data QUEUED for transmission		LINKING	*none*
CONT_WAIT		no data for transmission		SENSE	*none*

Detection of DCD (i.e., detection of a busy channel) while in the CONT_WAIT state is taken as an indication that the node has 'lost' the contention round to another, and thus, if the node still has data to transmit, it **shall** restart the LBT_WAIT_TIMER with value of LBT_WAIT_TIMER_DCD and transition to the LBT_WAIT state to wait once again for an idle channel.

Expiration of the CONT_WAIT_TIMER while the channel remains unoccupied (i.e. without any detection of DCD) is taken as an indication that the node has 'won' the contention round and thus, in this case and if the node still has queued data for transmission, the node **shall** transition to the LINKING state.

At any time while in the CONT_WAIT state, detection of an empty data queue (e.g., it may have been emptied by a management function) **shall** force a transition to the SENSE state.

K.3.2.5 CAS Linking State (LINKING)

Outbound transitions from the LINKING state shall conform to the table below.

Table 5 — LINKING-State Outbound-Transition Table

state	event	condition	action	next state	start timer
LINKING	(TX_BUFFER_EMPTY) or (TX_TIME >			SENSE	*none*

A node in the LINKING state transmit until its transmit-buffer is empty (i.e., TX_BUFFER_EMPTY == TRUE) or until the time it has transmitted exceeds the maximum transmit interval allowed to it (N.B.: a node's maximum transmit interval can be limited by various considerations, whether it is the 127.5 second limitation imposed by the EOT field size, or the time it takes to transmit the maximum number of unacknowledged DPDUs allowed by the ARQ protocol, or some other lesser time imposed by the subnetwork management function.

K.4 SETTING SCALAR CONTROL PARAMETERS

Operation of the CSMA protocol is controlled by the scalar parameters listed in the table below.

Table 7 —Default values for CSMA Scalar Parameters

Parameter Name	Default Value	Units	Default-Value Name; Comments
CONT_SLOT_WIDTH	3	seconds	CONT_SLOT_WIDTH; optimization of this value requires that it be a function of the modem preamble duration, data-rate, interleaver duration.
NUM_CONT_SLOTS	16	integer	For a slotted configuration, NUM_CONT_SLOTS should be set to the number of nodes on the network. For Jitter; the value selected is a balance between probability of one and only one node selecting the winning slot and acceptable delay (eg. 3 nodes, 90%probability, 16 slots).
LBT_WAIT_TIMER_VALUE_DCD	30	seconds	This needs to be set to a long value as the measurement may be premature due to fades
LBT_WAIT_TIMER_VALUE_VDCD	3	seconds	A short time is recommended to allow for potential errors in EOT calculation. This also needs to be long enough to give nodes waiting for this sufficient time to detect a node that responds immediately.

K.5 Changes since Edition 4

Edition 4 introduces a number of new timers and scalars, to meet the goals set out in Section K.2.1, which extend the capabilities of Edition 3.

Because Annex K requires that all nodes are configured consistently, this does not introduce interoperability issues. A deployment with a mix of Edition 3 and Edition 4 systems will need to be configured with Edition 3 settings only.

Status is changed from Informative to Optional. As Annex K controls timers it impacts interoperability and is not merely Informational. This change also brings it in line with Annex J.

ANNEX L HIGH-FREQUENCY WIRELESS-TOKEN-RING-PROTOCOL (WTRP) REQUIREMENTS

L.1. INTRODUCTION

This Annex specifies a High-Frequency Wireless Token Ring Protocol (WTRP) for enhanced media access control within single-frequency multi-node radio networks using STANAG 5066. The protocol is in two parts: a message design for the management tokens exchanged by nodes in the radio network, and the algorithms used to create, maintain, and repair the radio-transmission sequence (i.e., the virtual ring) of nodes in the network.

There is a token that represents right to transmit, which is transferred using a STANAG 5066 EOW (Engineering Order Wire) message. This means that the token can be transmitted without any protocol overhead and can be repeated for resilience. Additional messages to manage the token ring are sent using STANAG 5066 EXTENSION D_DPUs as specified in Annex C.

The High Frequency Wireless Token Ring Protocol (WTRP) is a self-organizing Medium Access Control (MAC) protocol for HF wireless networks. The MAC protocol by which mobile stations can share a broadcast channel is crucial in wireless networks, especially for HF wireless networks where bandwidth is considerably lower than other types of wireless networks. WTRP is a MAC protocol tailored for low-speed wireless networks. WTRP ensures that each node gets to transmit at least once per ring traversal, preventing the lock-out that can occur with CSMA. WTRP is efficient in reducing the number of retransmissions due to collisions.

All stations are connected on a single channel with common waveform operating on the same frequency. For waveforms with variable bandwidth a common bandwidth must be used by all stations. The frequency and bandwidth may be fixed or may be negotiated by ALE for all stations, as set out in Annex J. For autobaud waveforms, transmission speed and interleaver may vary during operation or fixed values may be configured.

WTRP requires that stations in a ring take turns to transmit for a specified maximum amount of time, with an order of transmission that includes each node at least once. WTRP is robust against single node failure. WTRP is different from its parent protocol in that it provides the notion of self-rings, that it supports connected networks with arbitrary connectivity, that there is no ring owner, and that stations can transmit more than once per ring cycle.

This Annex of STANAG 5066 is organized as follows.

- Section L.2 gives an overview of WTRP;

- Section L.3 specifies the STANAG 5066 message design for WTRP management messages;
- Section L.4 gives the complete WTRP state diagram and description for each state in WTRP;
- Section L.5 specifies parameters and timer selection for ring-management and operation;
- Section L.6 specifies requirements for token and message transmission.
- Section L7 summarizes changes since Ed3.

L.1.1. **Changes in This Edition**

The changes since Edition 3 are set out in Section L7 and repeated here for convenience.

The overall model and service provided by Annex L is unchanged. The state machine has some small changes. The protocol in Edition 4 is completely different to Edition 3 and interoperability is not a goal. This change was made because of significant problems with the protocol in Edition 3.

L.2. **OVERVIEW: WIRELESS TOKEN RING PROTOCOL**

Token-Ring definitions and management concepts are introduced below prior to detailed specification of the protocol in later sections. The original design of WTRP was based on work by Mustafa Ergen, Duke Lee et. al., “Wireless Token Ring Protocol”, University of California, Berkeley, CA 94720, USA.

L.2.1. **Definitions**

The following terms are used in the specification of the High-Frequency Wireless Token Ring Protocol (WTRP).

L.2.1.1. **Stations and Nodes**

The terms “station” and “node” are used interchangeably to describe the communication entities on the shared transmission medium.

L.2.1.2. **Token, Messages, Rings and Ring Members**

The WTRP protocol is a Medium Access Control (MAC) protocol. The task of this protocol is to schedule the access of two or more stations connected to the same physical medium, nominally an HF wireless network. Messages exchanged between stations for WTRP control are called *messages*.

The WTRP protocol organizes stations in such a manner that they rotate a *right-to-transmit Token (*or just *token)* among stations connected to the same physical medium. Only a *station* that receives the *right-to-transmit token* has the right to transmit user data. This *station* is then the *token-holder*. In normal operation there should only be one *token-holder*.

A set of stations sharing the same *right-to-transmit token* is defined as a *ring*, or *virtual ring.* A station participating in a ring is called a *ring member*.

When a station starts initially it is not a member of any ring. It will only be able to become a *ring member* if there is at least already one station connected to the same physical medium. If the station connected to the medium finds out there is no existing ring, it will attempt to set up a new ring with itself as its only member. Such a ring is called a *self ring*. If the station trying to establish a ring finds another station willing to join the ring, the *self ring* becomes a *ring*.

The *right-to-transmit* will be passed in rotation among the ring members, with a cycle that includes every ring member. A complete transit of this ring is referred to as a *ring cycle*.

L.2.1.3. Successors and Predecessors

If station S_A is passing the right to transmit to station S_B, then station S_B shall be called the *successor* of station S_A. The *predecessor* of station S_B shall be station S_A.

In other words, the *successor* of station X is the station to which X sends the *right-to-transmit token* to and the *predecessor* is the station from which X received the *right-to-transmit token*. Note that these stations are member of the same ring.

L.2.1.4. Ring Transmit Order

The order in which the Right-To-Transmit (RTT) token is passed around in the virtual ring is called the *transmit order*, which also defines the *successor* and *predecessor* relationship among *ring members*. For example, if the RTT token is passed from station S_A to station S_B, from station S_B to station S_C, and back to station S_A, then the transmit order is $\{S_A > S_B > S_C > S_A\}$. In a ring whose transmit order equals $\{S_A > S_B > S_C > S_A\}$, station S_B is the *successor* of station S_A, station S_C is the *successor* of station S_B, and station S_A is the *successor* of station S_C.

In a stable WTRP network, the *transmit order* will not vary and will include every *ring member*. In order to accommodate complex ring topologies, a node may appear more than once in the *transmit order*.

In a changing WTRP network, the node holding the token will choose its *successor* to optimize transmission, leading to a modified *transmit order*.

L.2.1.5. Node States

The following WTRP states are defined in the protocol:

- Floating State (FLT) - the initial (starting) state where a station is not part of a ring looks for an existing ring that it can join.
- Self Ring State (SFR) - in this state a station assumes there is **no** existing ring to join, and will therefore try to setup a new ring. The new ring will start with this station as the only member and is therefore called a self-ring.
- Seeking State (SEK) - in this state a station considers itself to be in a self-ring and has broadcast an INVITE message as an invitation for other stations to join its ring.
- Solicit Reply State (SRP) - in this state a station tries to join another ring; it intends to respond to a received INVITE message but is waiting for a timeout to avoid congestion before sending its reply.
- Joining State (JON) - In this state a station has replied to the invitation to join (i.e., to the received INVITE message) by sending a JOIN message to the node inviting it to join the net.
- Have-Token State (HVT) - in this state a station is part of a ring. It has received a RTT (right-to-transmit) token from its predecessor and with this token, the right to transmit. From this state a node may optionally transition to SLT in order to invite other nodes, after which it will return to this state. Then it will transmit token, messages and user data and transition to MON.
- Monitoring State (MON) - in this state a station is part of a ring. It passed the RTT token to its successor, but is unsure if the successor has received the right to transmit (i.e., the station is monitoring the channel for an implicit acknowledgement the RTT was successfully passed);
- Idle State (IDL) - In this state a station is part of a ring and knows it has successfully passed the RTT token its successor; it will process any D_PDU messages received from other stations;

- Soliciting State (SLT) - in this state a station is part of a ring, currently has the right to transmit, and is inviting new stations to join the ring by broadcasting an INVITE message and listening for replies;

L.2.1.6. Primary Timers

Most states have one or more associated timers. The primary Timers defined within the WTRP state machine are:

- Claim Token Timer (TCLT) - Timer used in the *floating-state* (FLT). Controls the time a station waits while in the *floating state* to claim a token before transiting to another state; a station restarts its TCLT timer when it transits to the FLT state.

- Solicit Successor Timer (TSLS) - This timer is used in the *self-ring-state* (SFR). If it times out the station shall transit to the *seeking-state* (SEK). The timer is specified with a random timeout to reduce the probability of collisions by transmissions from stations attempting to establish different rings at the same time.

- Solicit Reply Timer (TSRP) - Timer used in the *solicit-reply-state* (SRP). A station starts the *solicit-reply timer* when it transits to the SRP state. When it expires the station will send a JOIN message and transition to the JON state.

- Contention Timer (TCON) - Timer used in the *joining-state* (JON). It controls the time a station waits for a response from another station following an attempt to join the network, so-named because failure to receive a response is attributed to contention with other stations attempting to join the network at the same time; a station starts its *contention timer* when it goes to the JON state.

- Idle Timer (TIDL) - Timer used in the *idle-state* (IDL). It controls the time a station waits for its *right-to-transmit* before transiting to the *floating-state* (FLT).

- Solicit Wait Timer (TSLW) - This timer is used in the *soliciting state* (SLT) and the *seeking-state* (SEK) by stations inviting new ring members. When it expires, the inviting station will update the Transmit Order List to include all nodes that have sent JOIN messages.

- Token Pass Timer (TPST) - Used in the *monitoring-state* (MON). It controls the time a station waits after passing an RTT (or other) token to another *station* and failing to hear an implicit acknowledgement before considering the *right-to-transmit* as lost.

L.2.1.7. Token Contents

The right-to-transmit-token can be considered as a simple flag that controls the right to transmit. The token is transmitted along with information on the transmitting node's view of current ring size. This enables a joining node to better estimate ring latency and validates that ring members have a consistent view of ring size.

L.2.1.8. Promiscuous reception

Promiscuous reception is a receive mode in which a station performs limited processing and information collection on all D_PDUs it receives, whether or not they are addressed to it (i.e., the station takes in all traffic).

Promiscuous reception is the means by which a station discovers and confirms the existence or loss of links in the HF radio network (or that may be used to construct a network), compiles local node adjacency information, which is used to determine the choice of *successor*. This is core to the operation of WTRP.

L.2.1.9. Receive Table, Connectivity Table, and Next Hop Table

Each node maintains a *receive table* that records information on all other ring members where transmissions have been recently received. Each ring member will transmit regularly as the token passes around the ring, so it is certain that there will be transmissions from each node in a working ring. For each node from which a transmission has been recently received, the receive table will contain an entry indicating that this node can be heard and the quality of the reception. Where no transmission has been received from a node in the ring, the receive table will implicitly record that no data is being heard by not including the node in the receive table. While receive tables could be updated each ring cycle, it is generally desirable to make calculations based over a number of ring cycles (and reception from each other node) to avoid undue fluctuation of *receive table* values.

In addition, for each node where data can be heard, the *receive table* **shall** record a *transmit speed*. This speed reflects the maximum recommended speed for bulk throughput for data transmitted to the node. The model and encoding are aligned to the Data Rate Selection EOW encodings specified in STANAG 5066 Annex C. This speed will be calculated from the SNR and Frame Error Rate of data received from the node, using information obtained from the modem through the MAC layer.

Each node **shall** send its *receive table* to all other nodes using the TABLE PDU, by transmission around the ring whenever the receive table changes. This means that every node will have the *receive table* for all nodes, which provides the node with information on transmission of data to the node from every other node.

The *receive table* data from each node enables the node to build up an internal *connectivity table* for all nodes. For each node the *connectivity table* will record for each peer of that node:

1. The connectivity status of the node for each peer. One of:
    a. No direct connectivity
    b. Bidirectional transmission
    c. Transmission only from the node to the peer node.
    d. Transmission only from the peer node to the node.
2. Transmit speed from the node to peer node
3. Transmit speed from the peer node to the node

Unidirectional transmission can be useful for non-ARQ data. For ARQ data and token transfer, it is essential that transmission is bidirectional. Because of this, primary connectivity calculations in this specification are based on bidirectional transmission, which is safe for all types of transfer.

The *next hop table* is then built using information from the *connectivity table.* The next hop table is a key information that is passed upwards from WTRP, to enable higher layers to correctly route data.

The *next hop table* contains the following information for every node in the ring:

1. A choice of one of the following two connectivity options:
    a. The node can be reached directly for all transmissions; or
    b. The node can be reached directly for non-ARQ transmission only;
2. For nodes other than 1a, the preferred *data relay node* to be used for other communication.
3. Maximum transmission speed to be used for transmission to the node.
    a. Speed for direct transmission
    b. Speed for transmission to the *data relay node* (for nodes other than 1a)

Nodes with status 1a or 1b can determined directly from the *connectivity table*, and which also supplies the direct transmission speed. For nodes that cannot be reached directly, the *adjacency matrix* can be used to determine the best *data relay node*. For each directly connected node, the adjacency matrix can be used to determine paths to nodes directly connected to these nodes. This can be done iteratively to determine the distance to each node, and a set of paths to each node.

Where several directly connected nodes can act as the *data relay node*, one of them needs to be chosen for the *next hop table*. The usual choice of the preferred *data*

relay node is the one that is expected to receive the token next, which will usually minimize transfer time. The transmission speeds of each link **may** be considered, and it may be preferable to choose a *data relay node* that leads to faster transmissions on each hop. Note that the next hop table must be recalculated whenever *receive table* data changes.

The *next hop table* is used to communicate to the DTS and CAS layers of STANAG 5066, which nodes can be accessed directly, and which nodes need to be relayed.

To send data to nodes that are not directly connected, STANAG 5066 needs to send data to the *data relay node* for that node. The *data relay node* then needs to relay the data onwards to its final destination or to the next relay hop. The protocol to do this is specified in STANAG 5066 Annex R (Routing Sublayer).

L.2.1.10. Tour

A *tour* of a graph is a sequence of nodes from the graph such that each node appears at least once and two nodes are adjacent in the sequence only if they are adjacent in the *connectivity table*. An unconstrained, ordinary tour allows revisits to network nodes.

In the context of the wireless token-ring protocol, the sequence of ownership of the right-to-transmit (RTT) token, should be a *closed tour* of the network that starts and ends with nodes that are adjacent in the network, i.e., every node gets an opportunity to transmit, and passing the RTT token along the closed tour is feasible because nodes that are adjacent in the transmission sequence are adjacent in the network. A ring where the transmission order is stable is always a *closed tour* and may be an *unconstrained closed tour*.

L.2.1.11. Ring-Cycle Length (RCL)

The Ring-Cycle Length (RCL) is the length in hops of the tour through the WTRP network
taken by the RTT token as it completes a *closed tour*.

L.2.2. Concept of Basic Ring Operation

The basic concept of the Wireless Token Ring Protocol is to provide a mechanism that allows
two or more stations operating on the same single-frequency radio channel to share it in such a way that only one station will transmit at a time. Only the station that has the *right-to-transmit* (*token-holder*) is allowed to transmit on the shared channel. The *right-to-transmit* is passed onward to all stations that joined the ring.

Figure L-1 shows how the *right-to-transmit* is circulated among the *ring-members*.

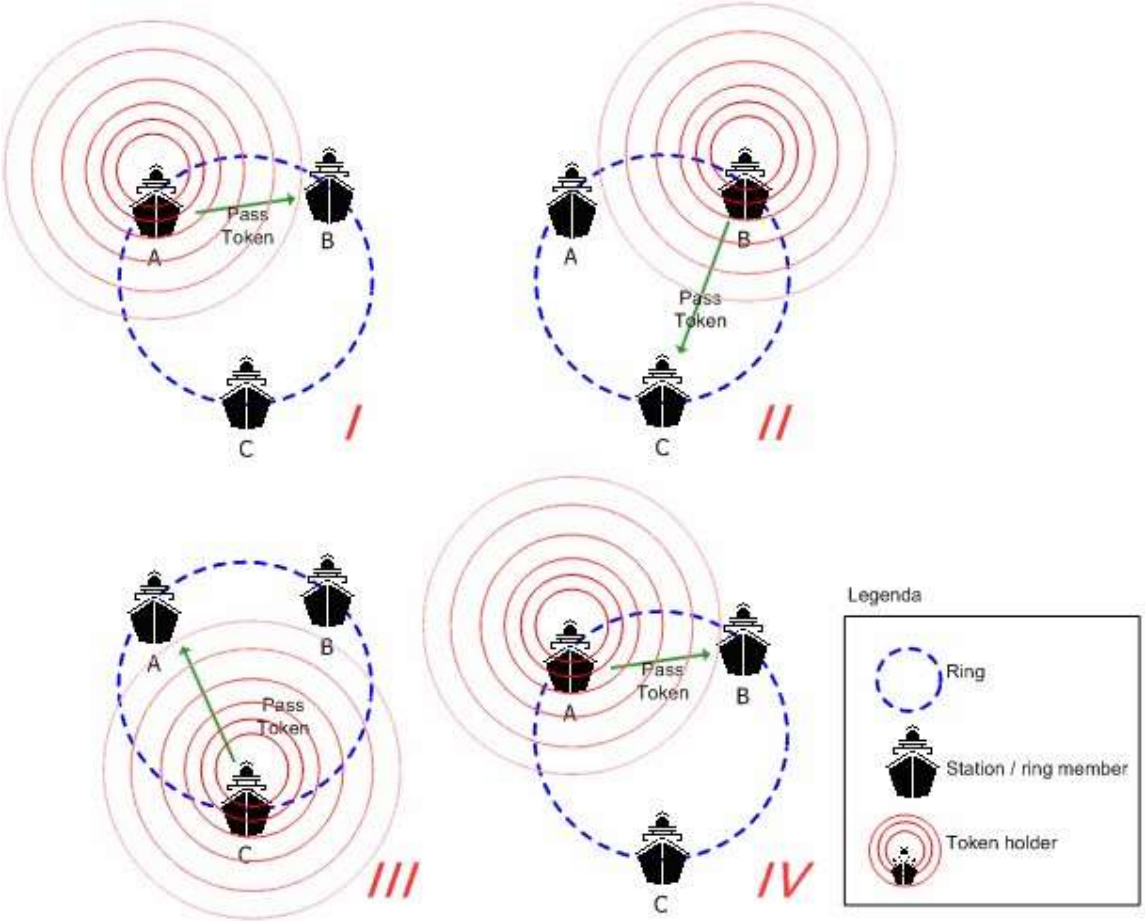


Figure L-1 - Normal Token-Ring Operation

The ring is a closed cycle of stations that transmit each in turn in a prescribed sequence, which will be adapted if new stations join the network or changes in network connectivity force adaptation. When a station receives the *right-to-transmit* from its *predecessor* it will take control of the channel. In addition to the *right-to-transmit token* the station may also receive additional WTRP messages, in particular updated *receive table* messages from one or more nodes. These messages will not be transmitted when the information is stable. *Receive table* messages are passed around the ring to ensure that all nodes have the latest information. A node will send its own *receive table* message if this information has changed. The *right-to-transmit token* is encoded in a standard STANAG 5066 EOW message and will generally be included multiple times in a transmission to minimize risk of token loss. If the station has no data to transmit, then it **shall** pass the *right-to-transmit* and associated messages immediately.

The messages sent by a station are transferred in a set of one or more D_PDUs. The D_PDU **shall** count down the *end of transmit time* (EOT) in accordance with the

requirements of Annex C. A station **shall not** exceed the maximum transmission time allowed.

Note that though the transmission sequence and ownership of the *token* in the ring is prescribed, a station with the *right-to-transmit* **may** send data to any other station in the network that it determines from the *next hop table* can receive data and transmit directly back acknowledgements for ARQ data, not just its successor or predecessor in the virtual ring. Any station in the virtual ring may therefore engage in multiple concurrent traffic exchanges with any other station in range. This includes the capability to support concurrent soft-link, physical-link, and ARQ connections in accordance STANAG 5066 Annex A, B, and C, with retransmission timers and other timing parameters selected to allow for the network size.

Possession of the *right-to-transmit* controls access to the radio channel; it does not prescribe the destination(s) to which traffic can be sent. The token-ring protocol controls node access to the transmission medium to preclude collisions and self-interference in the network and thereby increase the efficiency and throughput in the network. To do so, in general, there is no requirement that all nodes in the network be in communications range, only that the network have a closed tour, i.e., that a cyclic sequence of nodes exist where every node is in communications range of its predecessor and successor in the sequence. To support WTRP networks where direct communication is not possible between all nodes, the WTRP layer communicates connectivity to the higher layers of STANAG 5066. The higher layers will only send data to nodes that can be reached directly. Relay to other nodes is provided by STANAG 5066 Annex R (Routing Sublayer).

Each station will retransmit the *token,* passing the *right-to-transmit* to its *successor* until this has been acknowledged by its *successor.* This acknowledgement is implicit, by observing when the successor transmits data.

L.2.3. **Ring formation**

Before operating as a ring, a ring must be formed by stations on the same radio channel. The ring is a self-organizing and self-repairing mechanism, subject to a number of requirements:

- As soon as a station starts to operate it will listen to the radio channel in a floating state, unaffiliated with any ring.
- A station may transmit data (i.e., D_PDU other than ring-management messages) only if it is part of a ring that is not a self ring.

- A station shall first listen for an existing ring on the radio-channel; if it hears a ring it will try to join that ring, otherwise it will form a new ring (a self ring).
- If a station receives the right-to-transmit from another node, it has become a part of the ring.
- A station dropped from a ring must re-join the ring to become again part of it and obtain transmission rights.

There are two possible ways to form a ring:

1. Join an existing one.
2. Start a new ring yourself.

The process of Ring Formation is illustrated below in Figure L-2 - Ring Formation.

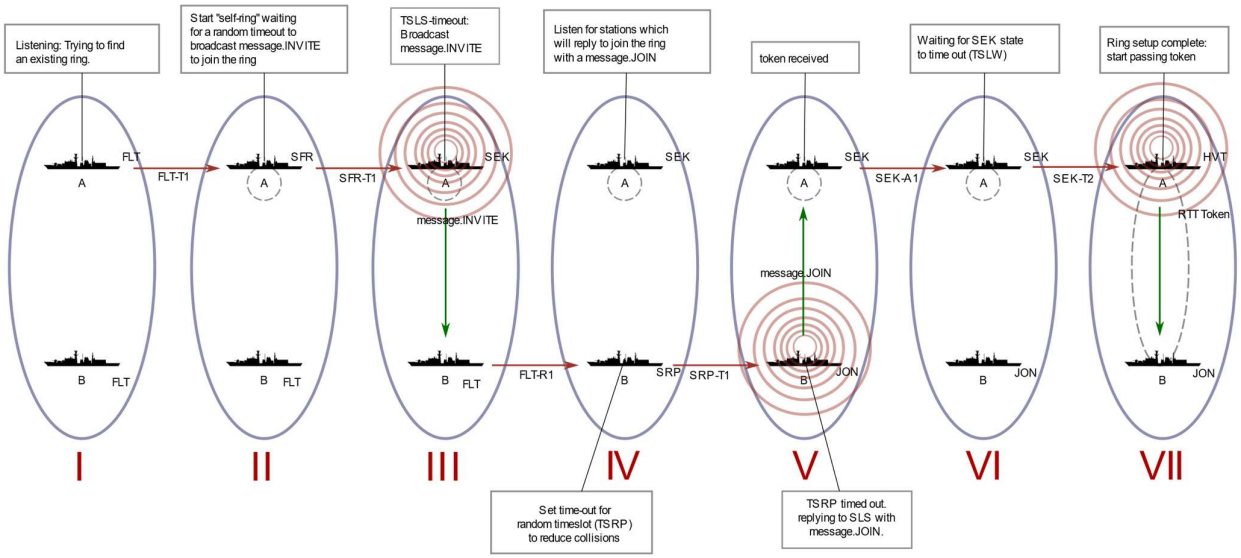


Figure L-2 - Ring Formation

Figure L-2 shows a time-sequence evolution of two stations, station A and station B, as they form a two-node ring.

In situation ***I*** station A is listening for an existing ring. After its TCLT timer times out, station A assumes there is no existing ring (otherwise it would have heard it) and it transits from a FLT state to the SFR state, forming a *self ring* as illustrated in situation ***II***.

The transition moment from the SFR state in situation ***II*** to the SEK state in situation ***III*** is controlled by a random timeout that avoids collision between two stations in the same state. The value of the timeout is calculated based on picking an asynchronous timeslot (See TSLS timer details in section L.5).

In the SEK state of situation ***III*** a station invites new members by sending an INVITE message. Each station that is seeking to join will reply with a JOIN (join ring) message. There could be more than one station waiting to reply to an INVITE message. Therefore, to reduce the probability of collision between replies from multiple joining nodes, the joining node's reply shall be transmitted in a randomly chosen time slot among a set of slots available for replies. All nodes in the ring and each node that sends a JOIN **shall** listen for JOIN messages so that nodes other than the one sending the INVITE can add nodes to the ring. The station that sent the INVITE message will add one of the nodes requesting to join to the Ring by passing the token to it. Other nodes that responded to the invite **may** be added by other nodes in the ring or by the node that sent the invite when the token returns to it.

Once the ring is created, each node periodically creates an opportunity to invite new stations to join the ring by broadcasting an INVITE message and waiting for JOIN message replies from nodes wishing to join the ring.

L.2.4. **Ring Entry**

The invitation to join shall be made periodically by a station receiving the right-to-transmit, with invitation frequency controlled by the algorithm specified in Section L.4.9. The opportunity to invite new members needs to be done by all ring members, as new nodes may only be able to connect to one existing member. A configurable maximum ring size **may** be specified, which can be used to stop rings growing too large. This maximum ring size **may** also be used when the number of nodes is known, to prevent taking time to invite new members when there are no possible new members. Invitations can also be configured so that they are less frequent (lower overhead) for stable rings.

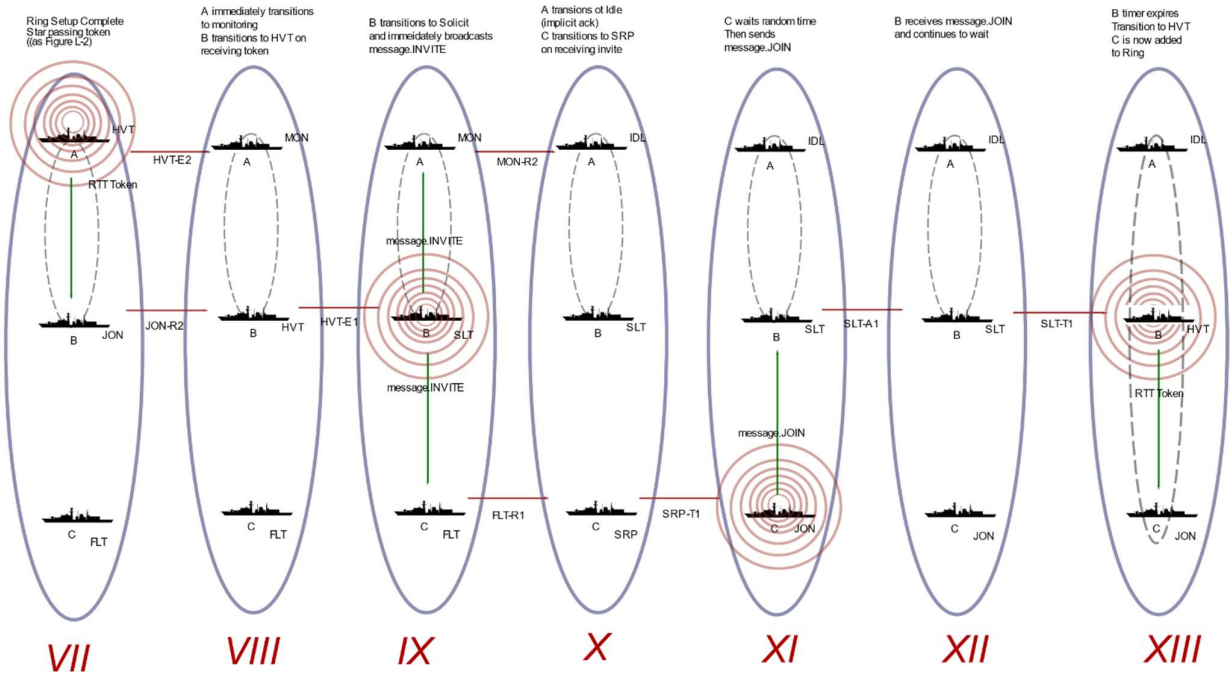


Figure L-3 - Ring Entry

Figure L-3 illustrates adding a station to an existing ring and continues the example of Figure L-2, showing addition of station C to the ring.

In VII, station B has received the token as in HVT state. Station B determines that it needs to issue an invite, so immediately transitions to SLT (IX) and immediately broadcasts a message.INVITE. Station A receives the message.INVITE, which it uses as implicit acknowledgement of token transfer, so it transitions from MON to IDL (X). Node C in FLT plans to accept the invitation and so transitions to SRP (X).

> NOTE: If there are multiple responses to the initial Invite, Station A **shall** respond to one of them. Other nodes **may** then respond to other responses.

Station C waits for a random time before responding with message.JOIN (XI). Station B receives the message.JOIN (XII) and continues to wait for potential other message.JOIN from other stations. After a timer, station B transitions to HVT and station C is added to the ring. Station B sends the token to station C, (XIII) and station C will enter HVT state after this.

L.2.5. **Connectivity Update**

Each node monitors the connectivity from all other ring members. This information

from other ring members is recorded in the *receive table*, which notes nodes that are heard and an associated recommended maximum transmit speed, determined from SNR, FER and other information.

Nodes will be removed from the *receive table* on either reaching a configurable maximum age or a configurable number of ring cycles with no data heard.

Nodes also monitor for transmissions from nodes not in the ring, which can lead to ring merging where a node is detected is in a different ring.

L.2.6. **Communicating Connectivity**

The calculated receive table is shared with all other nodes. This is done by sending a TABLE message that contains the receive table. TABLE messages are sent along with the token and will progress around the ring and will also be received by other nodes listening in promiscuous mode.

This information is important to optimize ring performance, but is not essential for the ring to operate. The TABLE update approach taken can lead to loss, which is seen as preferable to the additional overhead of a fully reliable mechanism.

TABLE messages are versioned. The model is that when a new version of the *receive table* for a given node is received it will be transmitted exactly once, unless this version of the *receive table* has been received from all nodes to which the node is connected.

Whenever the receive table changes, the version is incremented and a TABLE message sent to share this update with other nodes.

L.2.7. **Changing Ring Transmit Order**

There are four scenarios where the *transmit order* will change.

- Scenario 1 (Joining Scenario): A node joins the network and will be added to the ring.
- Scenario 2 (Leaving Scenario): A node chooses to drop out of the ring, due to operator request. In this scenario, the node does not transmit, leading to the token being transferred to other nodes and the node being dropped.
- Scenario 3 (Token Transfer Fail Scenario): Changes in the network may lead to a situation where token transfer to the intended current successor node fails.

This node will then choose a new successor to transfer the token to. If the token cannot be transmitted to any successor, the node drops out of the ring.

- Scenario 4 (Ring-Optimization Scenario): In this scenario a node determines a better order, with a different successor to the one previously used. To move to this new order, a node will transfer the token to this new successor.

L.2.8. Service Provision to Higher STANAG 5066 Layers

STANAG 5066 Annex J defines the MAC model which WTRP provides. There is a clean layer model, with M_ service elements communicating between the layers. The details of these service elements are not specified, but the model and associated protocols are clear. WTRP provides some additional services to the generic MAC functions defined in Annex J.

L.2.8.1. Data Rate, Interleaver and Transmission Length

STANAG 5066 DTS layer will control selection of transmission speed, interleaver and transmission length. The DTS layer will also determine which D_PDUs to duplicate, such as sending ACKs multiple times to improve reliability. WTRP provides information to DTS to facilitate this choice.

WTRP introduces new D_PDUs with rules for transmission and duplication. Selection of data rate, interleaver and transmission length must be cognizant of this information. The Annex C Data Rate Selection mechanism remains the primary mechanism for selecting speed. The *next hop table* defined in L.2.1.9 **may** be made available to the DTS to provide maximum transmission speed information for each directly connected node. This information will generally be consistent with the DTS Data Rate Selection information, but could given additional data.

L.2.8.2. Node Availability and Routing

WTRP determines which nodes in the ring can be accessed directly and for those nodes which cannot be accessed directly it specified a preferred relay node. This information is generated in the *next hop table*, as specified in L.2.1.9. This information is passed upwards as an Annex J M_ service primitive. This information can be used:

1. By the CAS as specified in STANAG 5066 Annex B to immediately reject messages to nodes that are not in the ring; and
2. By the Routing Sublayer as specified in STANAG 5066 Annex R to relay data to nodes that are not directly connected.

L.2.8.3. Broadcast Retransmission

When a broadcast/multicast PDU is received, information is provided to the higher layers. It is important that broadcast and multicast work correctly for nodes that are not directly connected. This needs to be done in a way that prevents duplicate onward broadcast. Consider four nodes connected in a long trapezoid shape, where the two end nodes cannot communicate directly. If one of the end nodes sends a broadcast message, then one (but not both) of the middle nodes that can communicate with both nodes needs to broadcast the message, so that all nodes receive it.

This is achieved by providing a *Relay Responsible List* to the higher layer for any broadcast or multicast PDU that is received. This tells the higher layers which nodes it needs to re-broadcast to. Handling this is described in STANAG 5066 Annex R (Routing Sublayer), which makes use of next hop table information to correctly.

The *Relay Responsible List* is used by the receiver of a multicast/broadcast PDU to determine which nodes it should relay to, in order to prevent duplicates and loops. There needs to be such a list for each connected node, as the calculation depends on the node broadcasting. All nodes have the same base *receive table* data, and so can perform the same calculation.

To determine the *Relay Responsible List* for a WTRP broadcast sender, take the following steps:

1. Consider each node in the ring. This is the starting list.
2. Determine *Next Hop Table* for the broadcast sender to be used in the next two steps.
3. Eliminate all nodes which can be directly reached by the broadcasting node using non-ARQ.
4. For remaining nodes, if the local node is the preferred relay, add the node to the *Relay Responsible List*.

L.3. WTRP TOKEN AND MESSAGE SPECIFICATION

L.3.1. Node ID

Nodes are identified by a single byte ID (range 0-255). This limits the number of nodes in a ring to 256, which is expected to be higher than the practical limit. This Node ID enables the TABLE PDU, which will typically be used quite a bit, to be more compact. Externally, each node is identified by a variable length STANAG 5066 address (typically 3.5 bytes).

The Node ID will usually be the same as the last byte of the node's STANAG 5066 address, unless there is a conflict in the ring. This enables provisioning of addresses to be done in a way which avoids Node ID conflict.

Each Node will maintain a map between Node ID and STANAG 5066 address for use with TABLE PDUs. The mapping is derived from the set of TABLE PDUs; when distributed the TABLE PDU includes this mapping. It is possible that Node ID conflicts will occur, typically when merging rings. If a node detects that it's Node ID is being used by another node, it will assign a new Node ID and send an updated TABLE message as soon as it can.

L.3.2. **Token Protocol**

The token is transmitted using a standard EOW message. This enables the token to be transmitted along with standard data. EOWs comprise one byte type and one byte content and can be carried in any DTS D_PDU.

The EOW **shall** be sent in at least one D_PDU with destination address of the token recipient. The source address will always be that of the token sender. If any WTRP TABLE PDUs are being sent they can also carry this EOW. The EOW can also be sent with user data going to the correct destination. It is generally desirable to include the EOW multiple times. If there is no D_PDU with the correct destination, ACK-ONLY or PADDING D_PDUs can be used to carry the EOW with low overhead.

The EOW type is value 13.

The single byte value of the EOW content encodes information on number of ring members and ring cycle length. This can be helpful for new nodes to set appropriate timers, prior to receiving full *receive table* information. The encoding is specified in Figure L-1, with values as follows:

- Node Count encodes the number of nodes, with the value reflecting the total number of nodes in the ring minus one (1). So a value of 0 indicates as self ring (one member). A value of 15 indicates 16 or more nodes.
- Hop Delta is used to share the *ring cycle count*: the total number of hops. It is encoded as a delta relative to Node Count. So if Hop Delta is 0, it indicates a fully connected ring without any extra hops. For a ring that has more than 16 nodes, the Hop Delta is the *ring cycle count* minus 16. If Hop Delta is 15, then this is a minimum value.

	MSB 7	6	5	4	3	2	1	LSB 0
0	Hop Delta				Node Count			

Figure L-4– EOW Content Encoding

This encoding anticipates that the practical upper limit of node count is around 10 nodes.

L.3.3. **WTRP Messages**

WTRP uses a set of messages to communicate information. These use EXTENSION D_PDUs as specified in STANAG 5066 Annex C. The allowed messages and defined extension D_PDU numbers are specified in Table L-1:

Message	Description	Extension Number
INVITE	Invite other nodes to join ring	1
JOIN	Request to join ring	2
TABLE	Receive Table	3
TABLE+	Receive Table (Extended Form)	4

Table L-1 – WTRP Messages

The syntax of these messages is specified in the following sections, and procedures to use them are specified in Section L.4. All of these D_PDUs use D_PDU type of 13 to identify EXTENSION D_PDU, and the extension number is then used to distinguish between each WTRP message.

It is expected that messages will generally be short enough to fit within a STANAG 5066 header, but the TABLE will not always be. Because of this, two forms of the TABLE message are defined. Use of different extension number allows the format to be determined from the extension number in the D_PDU header.

L.3.3.3 INVITE Message

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 1						LSB
CRC	CRC on Header							
CRC								

Figure L-5 – INVITE Message

The INVITE message is used by a node with the token to invite other nodes to join. It is generally sent to the broadcast address, so that any listening node will receive it.

It may be sent to a specific address, which is referred to as a Targeted INVITE Message. This is used when merging rings, to indicate to the recipient that the sender has heard the node and is inviting a ring merge.

L.3.3.4 JOIN Message

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)			Size of Header (h)				
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 2						LSB
CRC	CRC on Header							
CRC								

Figure L-6– JOIN Message

The JOIN message is used by a node responding to an INVITE message, indicating that it wishes to join the ring.

L.3.3.2 TABLE Message

	MSB 7	6	5	4	3	2	1	LSB 0
0	D_PDU Type = 13				EOW Type			
1	EOW							
2	EOT							
3	Size of Address Field (m)				Size of Header (h)			
3+m	Source and Destination Address							
4+m	MSB	Extended D_PDU Type = 3						LSB
5+m	MSB	Sending Node Address						
8+m								LSB
9+m	MSB	Sending Node ID						LSB
10+m	MSB	Version						LSB
12+m	Table (max 18 bytes)							
H+m-2								
CRC	CRC on Header							
CRC								

Figure L-7– TABLE Message

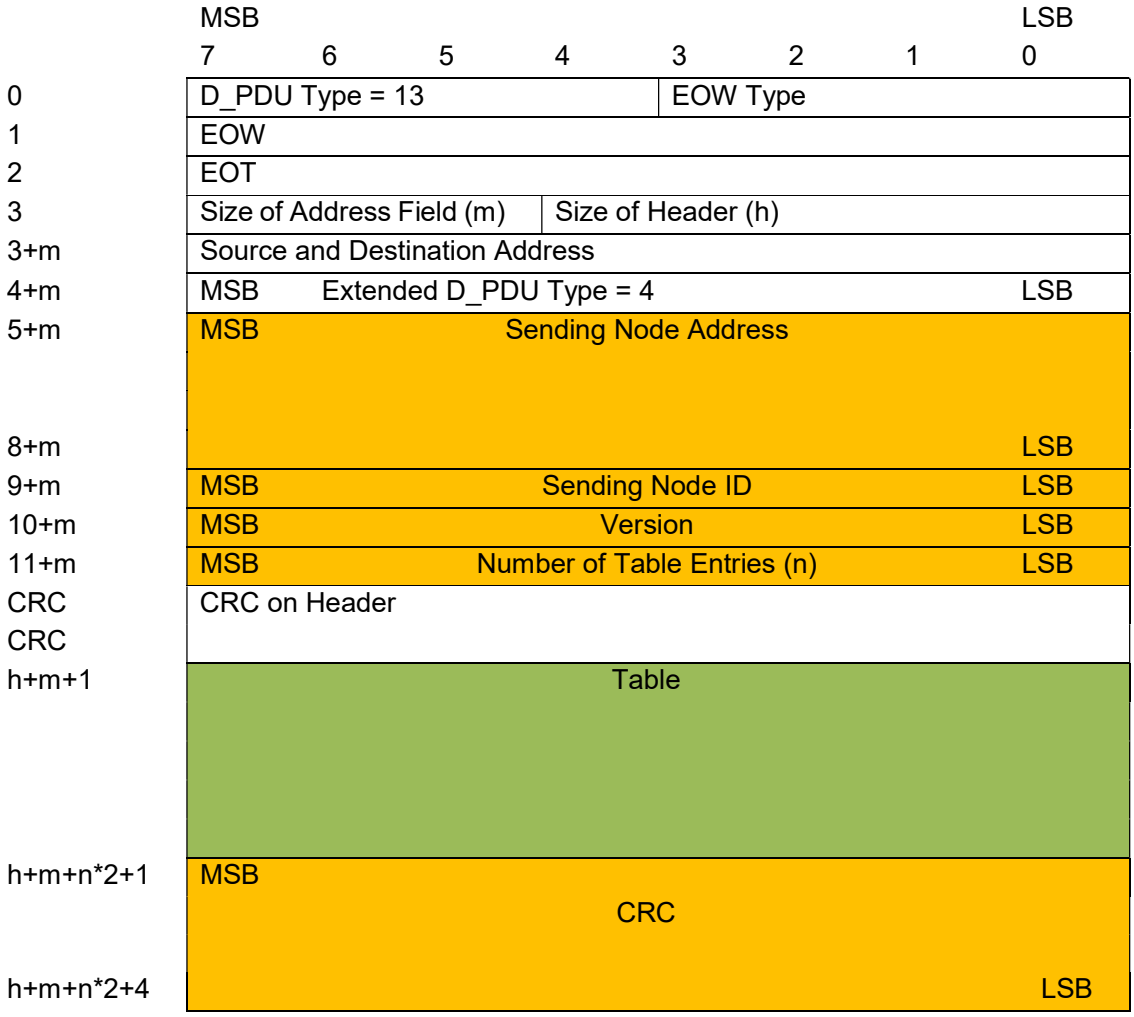


Figure L-8– TABLE Message (Extended Form)

The TABLE Message communicates a receive table. The Table is a sequence of two byte entries in arbitrary order. Tables of nine entries or less **shall** use the TABLE Message shown in Figure L-7. Longer tables **shall** use the TABLE Message (Extended Form) shown in Figure L-8. The fields are the same, except that the Extended Form has a Number of Table Entries header, which enables the length of the table, and thus the D_PDU length to be determined.

The Sending Node Address field is the STANAG 5066 address of the node that generated the receive table. The Sending Node field is the node ID of the node that generated the receive table.

The table has a version number starting at 0, which is incremented for each change, resetting to zero after 255. This is to ensure that when multiple versions of a receive table are circulating, that a node will ignore a version prior to the one it holds.

Each table entry comprises two bytes. The first byte is the node ID of the node that data can be received from. The second byte is the recommended transmit speed. This is encoded following the encoding specified in Section C.6.5.1 of STANAG 5066 Annex C (Transmission Speed and Interleaver Encoding).

L.4. PROCEDURE OF OPERATION

WTRP operation is defined in terms of the state machine specified here. Specifications and definitions are provided for states and state transitions. This makes use of a number of data structures and procedures defined in this section.

L.4.1. Node Data Structures

A node shall maintain the data structure listed Table L-6. These data structures are all prefixed with the following prefixes to facilitate clarity when they are referenced:

- "Node.": Generic information associated with the node.
- "LocalTable": Information associated with the local receive table.
- "PeerTables.": Information associated with receive tables from other nodes.
  .
- "Invite.": Information associated with control of invitations.

Table L-2 – Node Data Structures

Register / Flag	Type	Initial Value	Description
Node.address	STANAG 5066 Address	n/a	The station's address. The address is expected to be configured.
Node.ID	Integer	Last byte of Node.address	Node ID of the station. May be re-assigned in the event of conflict.
Node.cycleCount	Integer	0	The number of times that a complete ring cycle has been completed. This is used as an option to control sending invites only when the ring topology is stable.

Register / Flag	Type	Initial Value	Description
Node.Predecessor	STANAG 5066 Address	n/a	Set to sender of Token, when token is received.
Node.Sucessor	STANAG 5066 Address	*n/a*	Set to the Token receiver, when token is transmitted This is needed for monitoring after the token has been passed.
Node.TokenTransferred	Boolean	false	Acknowledgement is implicit, by monitoring traffic. This variable is set if traffic arrives that indicates token transfer
Node.OperatorDrop	Boolean	false	This variable is set by an operator. If set, it will cause the node to be dropped from the ring.
Node.txTokenCounter	Integer	0	Used to record repeats of token transmission.
Node.NextHopTable	Next Hop Table	empty	This is maintained so that higher layers of STANAG 5066 can select best speed for sending to a given node and can send indirectly to nodes that cannot be reached.
Node.NewNodes	List of Nodes with timestamps	Empty	Used to record JOIN requests, to build list of nodes to be added to ring. Associated with each node is a timestamp
Node.JoinsReceived	Boolean	False	Used to control ignoring of JOINs sent by other nodes
LocalTable.Table	Receive Table	Empty	The value of the local receive table, holding information on receive quality from peer nodes
LocalTable.TableChanged	Boolean	false	True if receive table has been changed since it was last transmitted
LocalTable.UpdateTimes	Array of Times, indexed by node address	empty	For each node in the ring, the time that the entry in LocalTable.Table was last updated.

Register / Flag	Type	Initial Value	Description
PeerTables.Tables	Receive Table List	empty	The active list of receive tables from peers. This list is updated whenever a more recent receive table is received from any node.
PeerTables.Transmitted	Received Table List	empty	List of receive tables indicating the most recent version that has been transmitted. This enables "transmit once" control.
Invite.LastTime	Time	Current time	When the last invite was sent by current node
Invite.Number	Integer	0	Number of invitations issued since joining ring

L.4.2. **Message and Token Notation**

The notation specified in this section is used to refer to tokens and messages and their components. Tokens have the following references, which are used to describe both reading and setting the token.

Table L-3 – Token Notation

Notation	Type	Description
token.sender	STANAG 5066 Address	Node sending the token from D_PDU source address
token.receiver	STANAG 5066 Address	Node to which the token is being sent taken from D_PDU destination address
token.inviterID	NodeID	The node ID of the node which is responsible for sending the next invite and for updating this field to set the subsequent inviter.

Messages are reference by notation of the form message.<message name>, for example message.JOIN. The following references to components of messages are used.

Table L-4 – Message Notation

Notation	Type	Description
message.<message name>.sender	STANAG 5066 Address	Node sending the message from D_PDU source address
message.<message name>.receiver	STANAG 5066 Address	Node to which the token is being sent taken from D_PDU destination address
message.TABLE.Table	Receive Table	Includes sender node ID and length

L.4.3. **Node Procedures**

The following functions are used in the state machine to specify node behaviour. Note that while this functional notation specified behaviour, this specification imposes no requirement on an implementation to use these functions.

---

isMember(<node address>)

Return type: Boolean

Description: Returns true if the given *node address* is a ring member, otherwise false.

Inspect Node.NextHopTable to determine if the specified node is present.

RingMemberCount()

Return type: Integer

Description: Returns the number of unique nodes in the Node.NextHopTable.

RCL ()

Return type: Integer

Description: Returns the Ring Cycle Length, which can be determined from the *connectivity table*.

BreakLink(<node address>)

Return type: None

Description: When a node cannot be reached, this is used to change network topology. This is used to recalculate the ring topology prior to determining a new successor.

This is achieved by finding the entry for <node address> in PeerTables.Tables and removing the entry for the local node in the identified receive table. This changes the connectivity record, to indicate that the node in question cannot be reached from the local node.

L.4.4. **Processing Inbound Transmissions**

This section, in conjunction with the following sections (L.4.5 – L.4.7) describes the process for handling inbound transmissions. This functionality is driven from the state machine by a single call of Receive(). This will listen for a call and continue processing until a full transmission has been received. At the end of transmission it will return EOT and optionally one of the following as events to the state machine:

1. EOT Event. To indicate end of transmission This is always returned, and the following associated Boolean is set:
    a. D_PDUs received. Set to true if any valid D_PDUs received.

    NOTE: EOT is returned at the end of every received transmission, irrespective of whether or not any D_PDUs are received.

2. One of the following may also be returned with the EOT event:
    a. Token. If a token directed to the local node has been received and none of the following messages.
    b. Message.INVITE. If this has been received, it will be the only message and no Token will be received.
    c. Message.JOIN. If this has been received, it will be the only message and no Token will be received. If this is received, the variable Node.JoinsReceived is set to true.

Processing of Message.TABLE reception is handled by this procedure and transparent to the state machine.

When a transmission is received it is fully processed, prior to any actions being taken in the state machine. D_PDUs other than WTRP messages are processed following the rules of STANAG 5066. If no other event is returned, EOT is returned at end of transmission.

The following WTRP messages may be received:

1. WTRP Message.INVITE is usually sent to the broadcast address, except for a Targeted INVITE that is sent to a specific address. A transmission with a message.INVITE may contain multiple copies of this message. Message.INVITE is returned to the state machine, when the target is a broadcast address or the local address.

2. WRTP Message.JOIN is sent to a single address. A single transmission may contain multiple copies of this message. Note that although Message.JOIN is directed to the inviter other nodes in the ring will also process it. If Message.JOIN.receiver is determined not be a member of the ring, the Message.JOIN is discarded. Otherwise Message.JOIN.sender is added to the Node.NewNodes list.

The sender of the transmission can be identified from any D_PDU in the transmission. Use of isRingMember(sender) determines if the sender is in the current ring. If the sender is not a ring member, the procedure of L.4.10 is followed.

Once a transmission has been processed, it will be possible to determine either the target destination node for the WTRP information, or that the message is a broadcast message.INVITE.

If the target destination node is the local node, the Section L.4.6 is followed. If the target destination is another node then Section L.4.5 is followed.

L.4.5. **Handing Transmissions Directed to Other Nodes**

Transmissions directed to other nodes are not directly handled by the state machine, but may lead to setting of variables that impact the state machine. WTRP must listen for these transmissions (promiscuous mode). Information in these transmissions is used to update local information.

Message.JOIN and Message.INVITE are special transmissions with no user data. Handling these is covered in Section L.4.4.

A message containing a token directed to another node indicates explicitly that the transmission is directed at another node. WTRP messages will always contain a token, so it will be always be possible to determine where a transmission is directed when it contains WTRP messages.

Transfer of tokens **may** be noted. Token.sender and token.receiver may be useful to provide as operator information to monitor progress of the token around the ring.

If the transmission sender of any D_PDU is Node.Successor, set Node.TokenTransferred to true. This setting is used to change out of MON state.

If a monitored D_PDU contains the WTRP token and isMember(token.Sender) is true, set Node.TokenTransferred to true. This setting is used to change out of MON state.

Received Message.TABLE messages are used to update local status. If the message.TABLE sender node ID is the local node, and message.Table.table matches the current or a previous version of LocalTable.Table, it is ignored. If it does not match, this means that another node is using the same Node ID as the local node. This is addressed by assigning a new Node.ID to one that is believed to be unique.

If isMember(message.TABLE.sender) is true the receive table is from the current ring. If it is not the local receive table, update the table in PeerTables.Tables with message.TABLE.table if the version is more recent.

Message.TABLE has version numbers encoded as a single byte. To compare version numbers of current and new, Mod(new – current, 256) < 127 will be true if new is more recent than current.

L.4.6. **Procedure for Receiving the Token**

This section describes how to handle a message with a token that is directed to the local node and therefore needs processing by the local node.

For each entry in LocalTable.Table look at the associated update timestamp in Local.TableUpdateTimes. If it is older than RECEIVE_TABLE_EXPIRY_AGE, remove the entry from LocalTable.Table and set LocalTable.TableChanged to true.

Next, Node.NextHopTable is calculated from the receive tables stored in PeerTables.Tables following the procedure specified in L.2.1.10. This information **shall** be passed up to the higher layers of STANAG 5066.

The Receive() procedure returns Token.

L.4.7. **Determining Successor**

This section sets out the approach for determining the successor, which is used prior to making a general purpose transmission.

The first step is to determine if there are any new nodes to be added to the ring. Node.NewNodes, which contains a list of nodes that have requested to join the ring is considered. First remove any nodes which are determined to already be in the

ring. If there are any remaining entries in Node.NewNodes, the oldest one **shall** be chosen as the successor and removed from the Node.NewNodes list.

The basic method of determining successor is by *connected node fairness*. This considers only nodes which can be reached directly. The best way to determine this list is from the *next hop table*, which gives a list of nodes with the two way connectivity needed for token transfer. If this information is not available, a list of nodes which are likely to work can be obtained from the receive table.

The *connected node fairness* algorithm is simply to send the token to the node that received it least recently. Because the local node can hear transmissions from all connected nodes, this is straightforward to determine.

It can be seen that the *connected node fairness* algorithm will work reasonably well for a fully connected ring. A partially connected ring can be considered as a concatenation of sub-rings, each of which are fully connected. It can be seen that this algorithm will ensure that all nodes are reached.

It is expected that the simple *connected node fairness* algorithm will lead to a reasonable ring choice in many scenarios, and in particular for many likely configurations.

This base algorithm is used, as it does not require any *receive table* derived information.

While this base algorithm is expected to be reasonable, it will not be optimal in all scenarios. Consider four nodes deployed in a square using HF surface wave, where optimum transmission speed decreases with distance. The best order will typically be to go around the edges of the square and avoid using the diagonals. The best ring cycle is likely to vary over time, as nodes move and conditions vary.

It is important to adapt to conditions. Consider a fully connected node network, with active transmission sequence ABCD. The *connected node fairness* algorithm will continue with this sequence, as long as all the nodes can communicate. If the best sequence was ABDC, it would need node B to choose to send to node D rather than node C (which is the one which has not had the token for the longest time).

A node can make use of the *connectivity table* to determine the best order. For a small set of nodes, it is practical to find the best order. For a larger set of nodes, determining best order is analogous to the well know “travelling salesman” problem, and seeking a good choice rather than best choice is likely to be computationally preferable.

NOTE: It is anticipated that operational experience will give feedback on the best approach to be taken, It is expected that this feedback will lead to updates to this standard.

L.4.8. **Sending Token, WTRP Messages and User Data**

A node will transmit data using the Transmit() procedure, which invokes the process described in this section. This is called from the state machine. At the end of transmission it will return an EOT-Transmit event to the state machine which indicates end of transmission.Transmission will be made to the node determined by following the procedure of Section L.4.7.

When a node has the token, it will make a transmission that includes the at least one copy of the token and may include user data. The token is encoded as an EOW and will usually be repeated many times in the transmission. Mechanisms to facilitate this are set out in L.6.

A transmission containing WTRP messages **shall** contain the token in each WTRP message. All WTRP messages and tokens in a transmission will have the same source and destination address. Other D_PDUs directed to the successor **may** contain the token. When token is transferred, it **shall** be put into at least one D_PDU. If necessary, a Padding D_PDU can be used for this. The token will generally be repeated many times.

The following WTRP messages **shall** also be transmitted. These messages may be repeated.

1. If LocalTable.TableChanged is true:
    a. message.TABLE is sent containing LocalTable.Table.
    b. LocalTable.TableChanged is set to false.
2. For each received table in PeerTables.Tables where the version is more recent than the one recorded in PeerTables.transmitted and the current version has not been received from all connected peers:
    a. message.TABLE is sent with the received table
    b. the version is updated in PeerTables.transmitted

User data, if available, **may** be sent in this transmission in addition to the messages above which **shall** be sent.

L.4.9. **Controlling Invitations**

This section specifies the algorithm for the node to determine whether or not to issue an invitation and setting the next inviter

Sending invitations needs careful tuning. There are a number of considerations:

1. Invitations should be sent from all ring members, as there may be nodes wishing to join the ring that can only hear transmissions from one ring member.
2. Invitations need to be sent sufficiently frequently so that new ring members can join in a timely manner.
3. In a stable long lived ring, adding new members will commonly be infrequent.
4. The invitation process is quite slow, and sending too many invitations will add unnecessary overhead.
5. In a partially connected network, it may be desirable to send invitations more frequently from the disconnected elements.

The approach taken in this specification is to have nodes independently choose when to send out invitations.

There is also an algorithm and a set of parameters specified which control how often a node issues an invite. This algorithm **may** be used or an implementation **may** choose a different algorithm as the exact algorithm will not impact interoperability although it can impact overall performance,

NOTE: When operational experience is obtained with the algorithm, a future version of this standard is expected to be updated to reflect this.

The algorithms is specified using two procedures that are called form the state machine.

ReadyToSendInvite ()

Return Type: Boolean

Description: Returns true if the criteria here are met

There is a configurable maximum ring size (MAX_NET_SIZE). If the number of notes in the ring is equal to or greater than this size, no invitation is issued. Procedure returns false.

The following parameters are used to control issuing and invitation:

1. Time since Invite.LastTime. If greater than MIN_INVITE_INTERVAL (configurable), an invitation **should** be issued, and procedure returns true.
2. Number of ring circuits since last invite by this node, determined by Node.CycleCount – Invite.LastCycleCount. If greater than or equal to MIN_INVITE_CYCLES (configurable), invite **should** be issued and procedure returns true
3. InviteNumber. If this is less than or equal to EARLY_INVITE_COUNT (configurable) number and ring circuits less than EARLY_INVITE_CYCLES (configurable), an invite should be issued and procedure returns true. This option **may** be omitted. It is designed to give a higher invitation rate in a new ring.

If none of the above conditions are met, no invite should be issued and procedure returns false.

SendInvite()

Return Type: None

Description: Send an Invite

Send a message.INVITE with:

- message.INVITE.destination = broadcast address

Update variables as follows:

- Invite.LastInviteTime set to current time
- Increment Invite.Number

L.4.10. Handling Transmissions from Nodes not in the Ring

Where a transmission and WTRP messages are received from a node not in the ring and not simply joining the ring, there are three possible scenarios identified.

1. A node forming a self ring that has not yet heard this ring. Strategy is to just let it find the current ring and join.

2. A node joining elsewhere in the current ring. This will sort out without any action.

3. Another formed ring. The approach is to merge the two rings. Care needs to be taken with rings which are on the edge of communication, because of potential instability due to poor links. This algorithm requires repeat hearing.

The definitive indication of another ring is token transfer. If this is not detected, the other transmission is ignored. If token transfer is detected, this will be recorded. If NUM_OTHER_RING_HEARD (configurable) of token transmissions are heard within OTHER_RING_TIME (configurable), this is considered definitive detection of another ring, which is within range.

When another ring is definitively detected, wait until the node has the token and then issue a Targeted INVITE, directed to the node that has been heard. This will signal to the invited node that it has been heard and also that no other nodes should respond to the invite. When the invited node gets the token, it will pass it over and the rings will be merged.

L.4.11. Overall State Diagram

Figure L-9 below shows the complete state diagram of WTRP. Each state is described in detail in the following sections. The starting state is Floating State (FLT). Note that most other states revert to FLT directly on error conditions, which is typically timer expiry. This figure shows all allowed transitions and marks the normal flow for a ring member. A node will be in Idle state (IDL) listening until the token is received. When the token is received, it will transition to Have Token State (HVT) where it will transmit the token and optionally user data. Then it will shift to Monitoring State (MON) where it listens to hear the next node transmit, which is an implicit acknowledgement of token transfer. When it hears this it transitions back to Idle State (IDL).

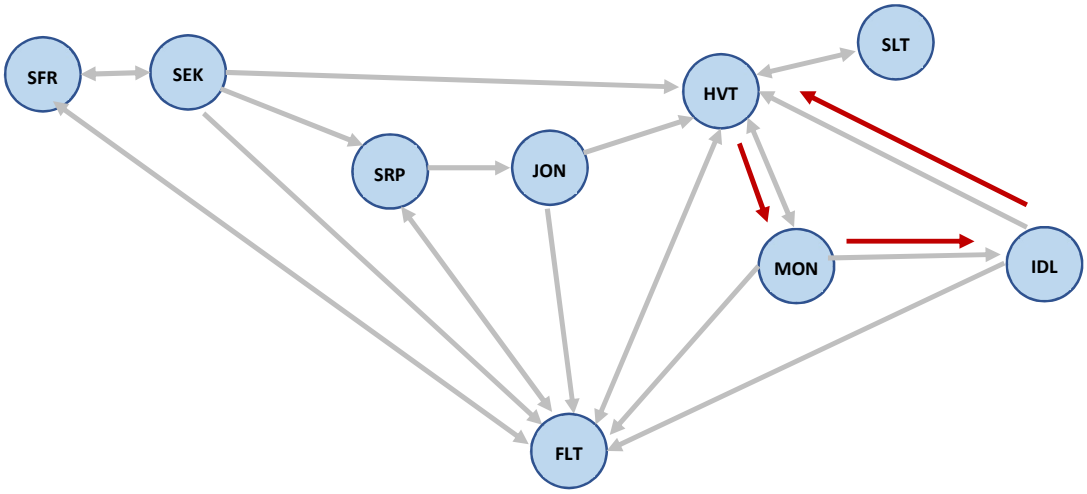


Figure L-9– Overall State Machine: Normal Ring Member Flow

Figure L-10 below illustrates how a node in an active ring will issue invites from time to time, breaking the normal state flow. This happens at intervals when a node is in Have Token State (HVT) it will, based on configurable rules, transition to Soliciting State (SLT) where it broadcasts and invitation to join the ring. It gathers any join requests which are then processed in Have Token State (HVT), either by this node or another one.

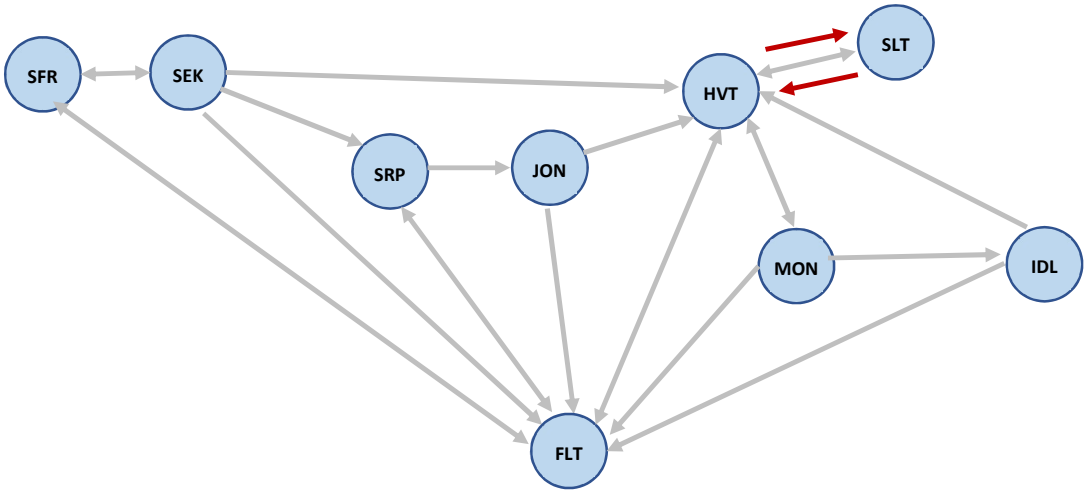


Figure L-10– Overall State Machine: Sending Invites

Figure L-11 below shows the normal path by which a node joins an existing ring. A node in Floating State (FLT) will hear a broadcast message invite. It will then transition to Solicit Reply State (SRP), send a Join message and transition to Join State (JON). The inviting node will send the token to the joining node which will transition to Have Token State (HVT) and become a ring member.

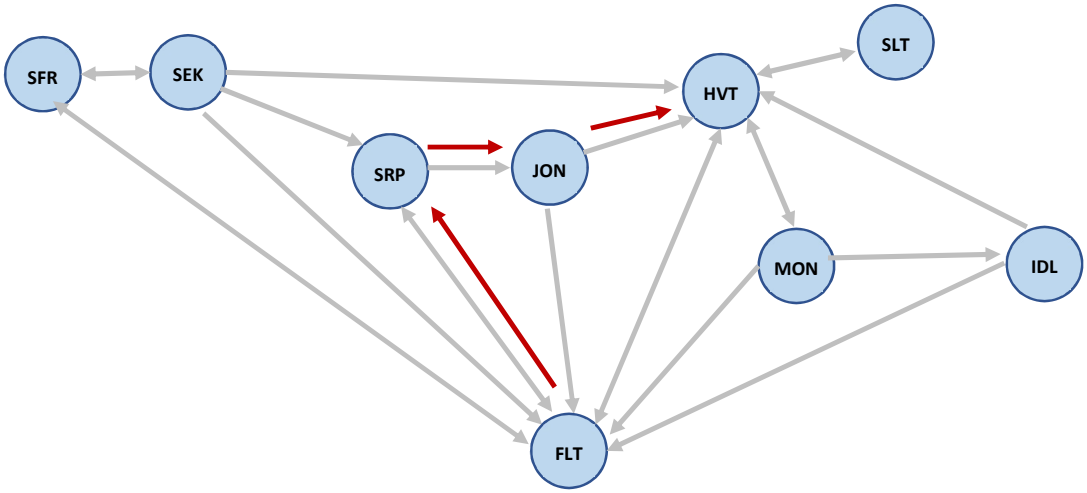


Figure L-11– Overall State Machine: Joining a Ring

Figure L-12 below shows the flow for a node initializing a new ring. A node starting in Floating State (FLT) will transition to Self Ring State (SFR). It will then broadcast a join invitation and move to Seeking State (SEK) where it waits for responses. When it gets one or more responses it will transition to Have Token State (HVT) and build the ring using the responses. It reverts to Floating State (FLT) if no other node responds.

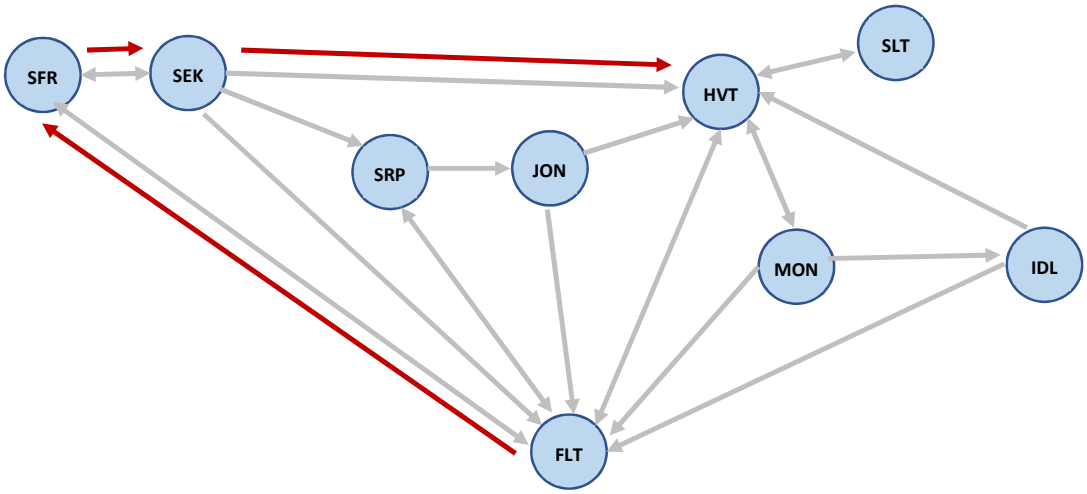


Figure L-12– Overall State Machine: Starting a New Ring

L.4.12. State-Machine Specification

This section and its subsections specify the actions of the WTRP state machine. For every state there are state-entry actions and an outbound-transition table defined. When entering a state, a station first **shall** execute the state-entry actions and then it **shall** wait for an event to occur which triggers one of the transitions to the next state defined in the outbound-transition table. There are two types of event that trigger state transitions:

- Events caused by timeouts; a timeout event is prefixed with the label “Exp” (i.e., for expiry or expired), followed with the name of the timer causing the timeout.
- Events caused by received data; this event is prefixed with the label “Rcv” (i.e., for ‘received’), followed with information to clearly identify what is received.

Only one transition rule **shall** be executed after an event: this **shall** be the first and only the first transition for which the condition is met as the state-machine logic examines the outbound-transitions in the order in which they are listed in the table.

When an action causes data to be transmitted, transition to the next state is expected to happen after the data has been transmitted and before any data is received.

L.4.12.1. Floating State (FLT)

The *Floating State* is the WTRP start-up state and the state in which a station is not part of a ring and waits to join a ring. The floating state is a *listening-only* state. A station **shall** stay in the FLT state until there is a joining opportunity (i.e., a message.INVITE is received inviting the station to solicit membership in the ring) or the TCLT timer expires.

The TCLT timer is used to determine when a station **will** assume that there is no existing ring present. If this timer times out (i.e., expires) the station **shall** proceed to *Self Ring State* (SFR).

If the station receives a message.INVITE it **shall** transit to the *Joining State* (JON) via the *Solicit Reply* (SRP) state,

L.4.12.1.1. Floating State entry actions

On this state entry the station **shall** execute the following actions:

- Start the TCLT timer
- Start Receive()

L.4.12.1.2. Floating State outbound transitions

Outbound transitions from the FLT state are shown in the table below. The most significant

outbound transition is to the Solicit-Reply (SRP) State, which occurs when a node receives an invitation to join a ring. For other transitions, the node either remains in the FLT state waiting to receive invitations or transits to the SFR state where it will wait before deciding to send its own invitations for nodes to join its ring.

If any transmission is heard, it will cause the node to wait longer by setting the TCLT timer to TCLT_TRANS_HEARD. TCLT will reset to TCLT_SILENT if no further transmissions are heard.

It is possible that the token will be passed to a node in FLT state. This can happen if node is in ring and transmits token successfully without hearing the onward transmission. In this situation, the node can move to FLT state (from MON) and then later receive a valid token.

Table L-5 - FLT outbound transitions

transition	event	Condition	Action	next state
FLT-R1	EOT	Message.INVITE	Save message.INVITE for processing in SRP state	SRP
FLT-R2	EOT		Set TCLT to TCLT_TRANS_HEARD Start TCLT Timer Receive()	N/A
FLT-R3	EOT	Token		HVT
FLT-T1	Exp: TCLT		SET TCLT to TCLT_SILENT	SFR

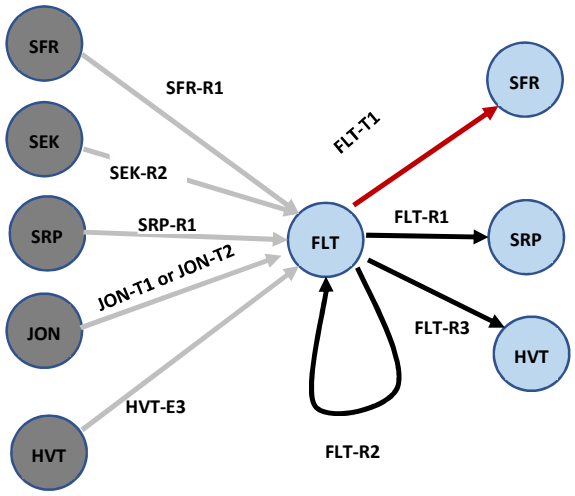


Figure L-13 - FLT Outbound transitions

L.4.12.2. Self Ring State (SFR)

In this state a station has concluded there is no ring to join and therefore will setup a new ring by itself. This condition is called the *self ring*. This state is like the Floating State (FLT) in that it is a listening-only state.

On state entry the TSLS timer **shall** be set to a random timeout value, which is used to avoid collisions with any other station trying to setup a ring. If this timer (i.e., if TSLS) times out the station **shall** transit to the *Seeking State* (SEK), where an invitation to nodes to join its network will be sent as an inbound action.

While waiting for the TSLS timeout, a message.INVITE **might** be received from another station in SEK state; in this case the station **shall** transit to the *Solicit Reply State* (SRP), where it will respond to the invitation.

If any other transmission is received the station **shall** go to the *Floating State* (FLT).

L.4.12.2.1. Self Ring State (SFR) entry actions

On SFR-state entry a station **shall** start its TSLS Timer with a random time-out value over a configurable range. Receive() procedure.

L.4.12.2.2. Self Ring State (SFR) outbound transitions

Outbound transitions for the SFR state are shown in the table below. Transitions from the SFR state are triggered: when the station receives an invitation from another node and then will reply; when the station receives any other D_PDU from another station and thus should wait for an invitation; or when the station hears nothing for some time and then sends its own invitation.

Table L-6 - SFR-State Outbound-Transition Table

transition	event	condition	action	next state
SFR-R1	EOT		Set TCLT to TCLT_TRANS_HEARD	FLT
SFR-R2	EOT	Message.INVITE	Save message.INVITE to be processed in SRP state	SRP
SFR-T1	Exp: TSLS		See L.4.9	SEK

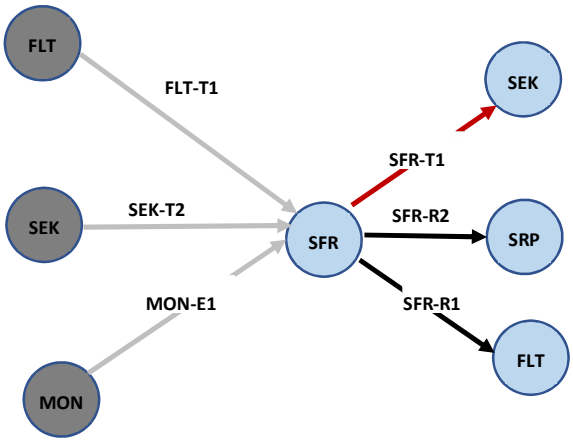


Figure L-14 - Outbound transitions from SFR state

L.4.12.3. Seeking State (SEK)

The *Seeking State* is a state in which a station in a *self ring*, the seeking node, broadcasts a message.INVITE inviting new ring members and listens for replies from solicitors until the TSLW timer times out.

When the TSLW timer expires and no message.JOIN has been received, the node reverts to FLT state. If one or more message.JOIN has been received, they will be recorded to enable addition to the ring. The station will then proceed to the HVT state.

If a message.INVITE is received before the TLSW timer expires, then another ring (possibly a self-ring) is active within radio range of the station. The station in this case **shall** cease its invitations to other nodes to join its self-ring and **shall** transit to the SRP state, with intent to join the ring it has detected.

Reception by a station when it is in the SEK state of any other transmission shall force the station into the FLT state, as reception of such tokens is an indication there is an active ring within radio range, the self-ring condition no longer applies and the station should not be soliciting to form its own ring.

L.4.12.3.1. Seeking State entry actions

On state exit from the SFR state the station has broadcast a message.INVITE, and shall start the TSLW timer waiting for replies. Start Receive() procedure.

L.4.12.3.2. Seeking state outbound transitions

Outbound transitions from the SEK state are shown in the table below. As noted above, reception of message.JOIN does not force an outbound transition; the message.JOIN messages are stored for later processing and the station remains in the SEK state. Reception of a message.INVITE forces a transition to the SRP solicit reply state and reception of any other transmission forces a transition to the FLT floating state; both of these triggering events invalidate the station's assumption that it can form its own ring but with different responses by the station. As long as the station receives only message.JOIN messages, it will continue to handle them until the TLSW timer expires, at which point the station proceeds to the HVT state and subsequent operation in a multi-node network. If no responses of any kind are heard after waiting for replies, the station returns to the SFR self-ring state.

Table L-7 - SEK state outbound transitions

transition	event	condition	action	next state
SEK-R1	EOT	Message.INVITE	Save message.INVITE for processing in SRP state	SRP
SEK-R2	EOT	D_PDUs Received set to true	Set TCLT to TCLT_TRANS_HEARD	FLT

SEK-R3	EOT	Message.JOIN	Receive() Note that Message.JOIN is processed within Receive() following L.4.4	N/A
SEK-R4	EOT	D_PDUs Received set to false	Receive()	N/A
SEK-T1	Exp: TSLW timer	At least one message.JOIN received		HVT
SEK-T2	Exp: TSLW timer	No message.JOIN received		SFR

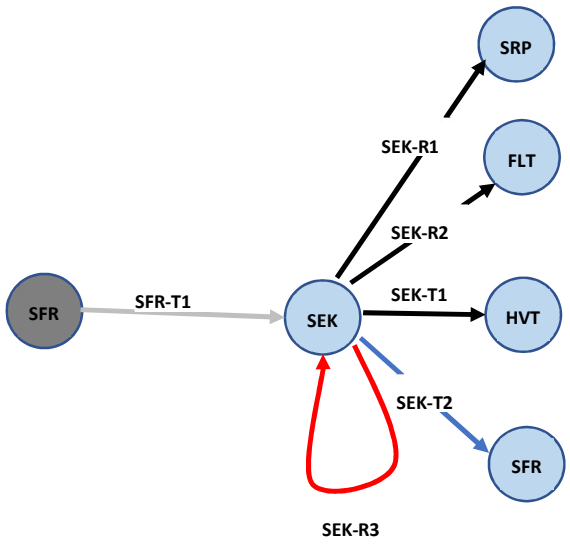


Figure L-15 - Outbound transitions from SEK state

L.4.12.4. Solicit Reply State (SRP)

In the *Solicit Reply State* a station will reply to an invitation to join the network (i.e., to a message.INVITE received from another node) and attempt to join the ring.

The TSRP timer **shall** be started on state entry to determine the moment to reply.

While waiting in the *Solicit Reply State* for the TSRP time to expire, other message.JOIN messages might be
received and should be ignored, as they could be expected as replies by other stations to the message.INVITE.

Receipt of other message traffic indicates that the channel is not clear during the protocol's invite-and-solicit dialog for new ring members (an error condition in the protocol). The error condition is best handled by having all solicitors return to a known state (the FLT state) in which they do not transmit.

The soliciting station **shall** reply by sending a message.JOIN to the inviting station, the message.INVITE originator.

L.4.12.4.1. State entry actions:

Set the TSRP timer following the algorithm defined in Section L.5. Start Receive() procedure

L.4.12.4.2. Solicit Reply State Outbound Transitions

Outbound transitions from the solicit transition can be categorized as error recovery or incremental success in joining the network. If any message other than a message.JOIN is heard, the station effectively declares error and transitions to the FLT state, where it will wait for another invitation to join. If the channel remains clear of unexpected traffic, the node sends its solicitation (a message.JOIN) to join the network and transits to the JON state where it will wait to see if its solicitation succeeded.

Table L-8 - SRP outbound transition table

transition	Event	condition	action	next state
SRP-R1	EOT	Node.JoinsReceived set to false and D_PDUSs Receives set to true (Message.JOINs being sent by other nodes in response to invite are ignored)	Set TCLT to TCLT_TRANS_HEARD	FLT
SRP-R2	EOT	(other than SRP-R1)	Receive()	N/A

SRP-T1	Exp: TSRP		Send message.JOIN where message.JOIN.sender = Node.address message.JOIN.receiver = address of inviter message.JOIN.Table = LocalTable.Table	JON

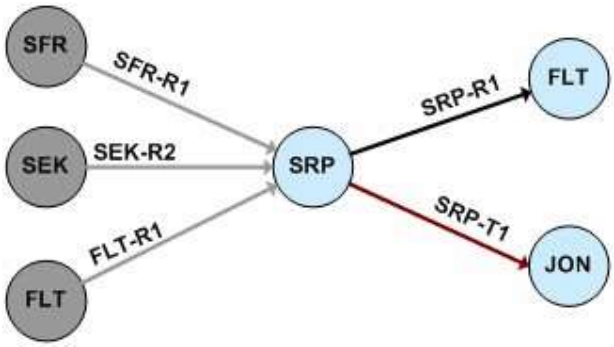


Figure L-16 - Outbound transitions from SRP state

L.4.12.5. Joining State (JON)

In the *Joining State* a station has replied to solicitation opportunity by sending a message.JOIN and is now waiting for the *right-to-transmit token*.
The test that the soliciting station has successfully joined the ring is straightforward; the soliciting station either receives an RTT token (in which case it joins the ring) or it does note.

L.4.12.5.1. Joining State entry actions

Start the TCON timer. Start Receive().

.

L.4.12.5.2. Joining State outbound transitions

Outbound transitions for the JON state are shown in the table below. A station either succeeds with its solicitation to join, and transits to the HVT state as a ring member and able to transmit, or fails and transits to the FLT state where it will wait for the

next invitation from a ring member (or its own declaration that there is no ring present and eventual transition to the SFR self-ring state).

Table L-9 - JON-State Outbound-Transition Table

transition	event	condition	action	next state
JON-R1	EOT		Receive()	N/A
JON-R2	EOT	Token		HVT
JON-T1	Exp: TCON timer		Set TCLT to TCLT_TRANS_HEARD	FLT

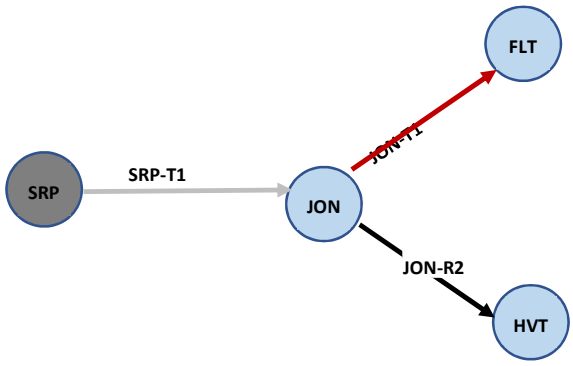


Figure L-17 - Outbound transitions from JON state

L.4.12.6. Have Token State (HVT)

In the *Have Token State* a station has received the *right to transmit token* (RTT) and as part of the state exit action the channel "owner", is allowed to send D_PDUs as long as the length of its transmission does not exceed the *maximum time to transmit*.

L.4.12.6.1. Have Token State entry actions

The node will have received the token in the previous state, which causes the node to enter HVT. The received transmission will have been processed according to L.4.6.

If map or receive table information is needed, but not received, token is sent to predecessor with message.RETRANS and node enters MON state.

The node will verify whether or not an invitation should be sent, following the procedure in L.4.9. If it is determined that an invitation should be sent, and invitation is sent and the node transitions to SLT state. After SLT, the node will return to HVT.

After any invitation has been sent, the token, optional messages and optional user data will be sent following L.4.8, and the node transitions to MON state. L.4.8 will also handle the following special situation:

1. The HVT state is where the node processes operator or other local requests for the node to drop out of the ring. The node will simply not transmit data and return to FLT state, ignoring when the token is sent to it.

L.4.12.6.2. Have Token State outbound transitions

Outbound transitions from the HVT state are shown in the table below.

Transitions from the HVT state to the MON state happens after the transmission has completed, which is indicated by the EOT-Transmit event The TEOT timer protects against missing EOT-Transmit event.

Table L-10 - HVT-State outbound transition table

transaction	event	condition	action	next state
HVT-E1	State entry	ReadyToSendInvite () == true	SendInvite()	SLT
HVT-E2	State entry	ReadyToSendInvite () == false && RingMemberCount() > 1	Set TEOT Transmit() following L.4.8	N/A
HVT-E3	State entry	ReadyToSendInvite () == false && RingMemberCount() <= 1	Set TCLT to TCLT_SILENT	FLT
HVT-R1	EOT-Transmit			MON

HVT-T1	Exp: TEOT			MON

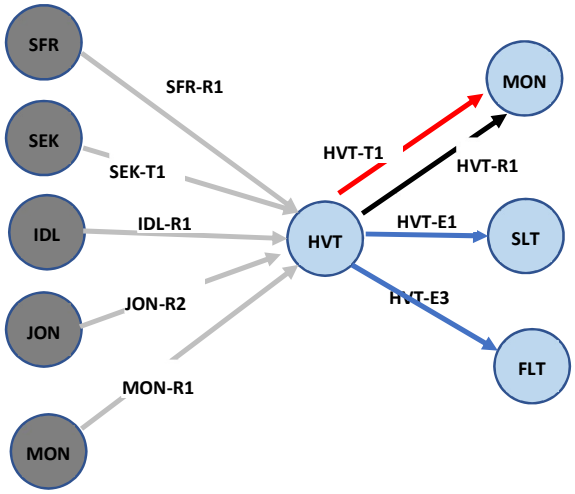


Figure L-18 - Outbound transitions from HVT state

L.4.12.7. Monitoring State (MON)

The *Monitoring State* is a state in which a station has finished transmitting data, passed the RTT token to its successor and is waiting for an acknowledgement. An implicit acknowledgement mechanism is used. Any D_PDU received in the MON state with source address equal to the station's successor are taken as implicit acknowledgement that the successor received the RTT token and has taken control of the radio channel. Also, any token transfer on the current ring indicates that the token is circulating in the ring. This second mechanism addresses when the transmission by the successor is not heard. This monitoring is handled by L.4.5 and reflected in the variable Node.TokenTransferred.

In a two node ring, the token will be passed directly back.

At expiration of the TPST timer the token transfer is considered to have failed. The station shall resend it for a number of times until the MAX_TOKEN_PASS count has been reached. It is recommended that retransmissions are done at progressively slower speeds.

If the MAX_TOKEN_PASS count has been reached the station declares the next-hop node unreachable and reflects this in connectivity by using the breaklink() procedure. The Transmit() procedure of L.4.8 is followed again. This process continues until the token is transferred or there is no possible node to transfer to. If all other nodes are eliminated, the node transitions to SFR.

L.4.12.7.1. Monitoring State entry actions

Transmission, including token transfer, will have been completed prior to state entry. On state entry the station **shall** start the TPST timer, setting the time the station shall wait for receipt of an implicit acknowledgement that the *right-to-transmit* has been transferred.

For any entry other than a self-transition, i.e, if the preceding state is not the MON state, Node.txTokenCounter is set to 1.

Start Receive()

L.4.12.7.2. Monitoring State outbound transition table

Outbound transitions from the MON state are shown in the table below.

Transitions representing a successful pass of the *right-to-transmit* token are to HVT state (token comes back) or to IDL state.

Other transitions will cycle through the MON state to transmit to nodes in turn. When a node is joining the ring, it will only attempt to transmit back to the node that sent the token to it.

Table L-11 - MON-State Outbound-Transition Table

transition	event	Condition	Action / Comments	next state
MON-E1	State entry	RingMemberCount() < 2	If ring has been reduced to 1, move to float Set TCLT to TCLT_SILENT	FLT
MON-R1	EOT	Token		HVT
MON-R2	EOT	Node.TokenTransferred == true	Implicit Ack Set Node.txTokenCounter to 1	IDL
MON-R3	EOT	Node.TokenTransferred == false	Receive()	N/A
MON-T1	Exp: TPST	Node.txTokenCounter < MAX_TOKEN_PASS	Increment Node..txTokenCounter Return to HVT to force transmission of token again.	HVT
MON-T2	Exp: TPST	Node.txTokenCounter >= MAX_TOKEN_PASS	Set Node.txTokenCounter to 1 BreakLink(Node.Successor) Return to HVT to force transmission of token again.	HVT

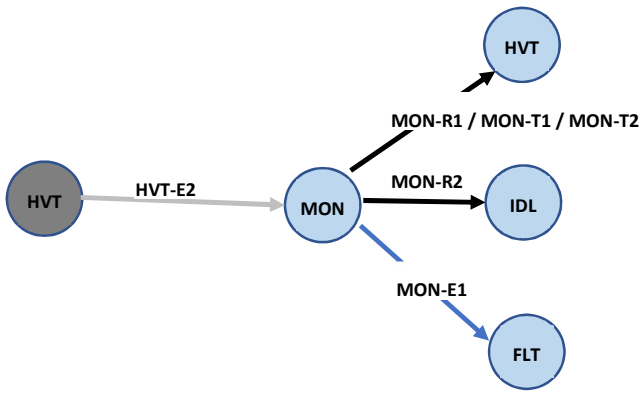


Figure L-19- MON Outbound transitions from monitoring state

L.4.12.8. Idle State (IDL)

In the *Idle State* the station waits for the *right-to-transmit token* until its TIDL timer expires. When it receives an RTT token, it transits directly to the HVT state. In the IDL state, the node listens for traffic following L.4.4, L.4.5, L.4.6 and L.4.10.

If the TIDL timer expires the station assumes an error-condition in the protocol and transits to the FLT state to recover.

L.4.12.8.1. Idle State entry Actions

The station **shall** start its TIDL timer on state entry. Receive()

L.4.12.8.2. Idle State (IDL) outbound transitions

Outbound transitions from the IDL state are shown in the table below.

Table L-12 - IDL State Outbound-Transition Table

transition	event	condition	action	next state
IDL-R1	EOT	Token		HVT
IDL-R2	EOT		Receive()	N/A
IDL-T1	Exp: TIDL		Set TCLT to TCLT_SILENT	FLT

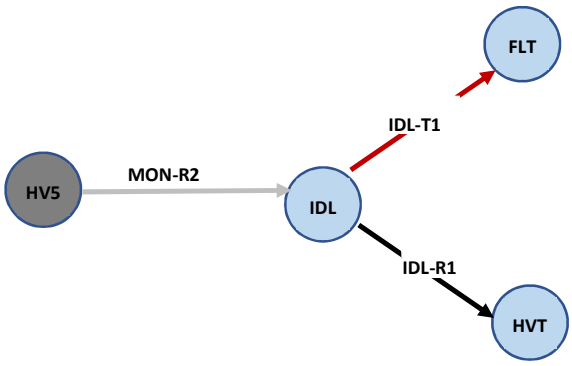


Figure L-20 - Idle State outbound transitions

L.4.12.9. Soliciting State (SLT)

The purpose of the SLT state is to allow new stations to join the ring. A station in the *Soliciting State* is a ring member and has received the *right-to-transmit* from the previous HVT state. A message.INVITE was sent on transition to SLT.

The station waits until TSLW timer times out and then returns to the HVT state. Every message.JOIN is processed, which will be used to determine the successor in HVT state.

L.4.12.9.1. Soliciting State (SLT) entry actions

On state entry the station **shall** start the TSLW timer and listen for message.JOIN.

Start Receive()

L.4.12.9.2. Soliciting State (SLT) outbound transitions

Outbound transitions from the SLT state are shown in the table below.

The normal transition (SLT-T1) to the HVT state occurs when the TSLW time expires, and is made whether or not new stations have solicited to join.

Table L-13 - SLT-State Outbound-Transition Table

transition	event	condition	action	next state
SLT-R1	EOT	Message.JOIN	Receive() Note that Message.JOIN is processed within Receive() following L.4.4	N/A
SLT-R2	EOT		Receive()	N/A
SLT-T1	Exp: TSLW			HVT

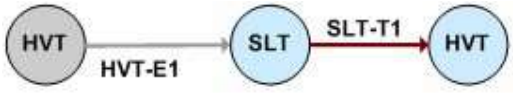


Figure L-21 - Outbound transitions from soliciting state

L.5. TIMERS AND PARAMETERS

This section reviews all of the parameters and timers. The core of this section is a table with all of the specified timers and parameters. The rationale for the parameter and considerations for optimal settings are described. Each of these parameters and timers **shall** be modifiable by an operator to change the protocol responsiveness or

behaviour in response to different operational requirements or tradeoffs (e.g., increased collision probability for newly joining nodes versus reduced solicitation overhead).

L.5.1. HF Considerations

HF provides a difficult channel with highly variable quality where the optimal transmission speed varies significantly, with short and extended periods where no communication is possible. The nature of this channel is a primary consideration in choosing parameter and timer values.

L.5.2. Surface Wave Considerations

A major target for WTRP is surface wave, particularly for communication between naval vessels where surface communication is good and extends a long way beyond line of sight. WTRP is less suitable for HF Sky Wave, where it will generally be preferable to use ALE to optimize point to point links, rather than attempting to use a single channel for many nodes. For this reason, default parameters are chosen on the basis of using surface wave to communicate between a group of ships. Where WTRP is used in other scenarios, it is likely to be desirable to change defaults.

With surface wave, SNR will systematically decrease with distance. This means that SNR will change as nodes move and will be better for nodes that are closer together. For this reason, it is anticipated that the variable speed operation provided by this specification will be highly beneficial to good performance.

At the limit of communication distance, communication is expected to be very intermittent. Care needs to be taken with this, as there is possibility to spend resources trying to maintain a link between a pair of nodes where they can hear each other, but the channel is too poor for robust transfer.

L.5.3. Ring Forming Strategy

A node that is not in a ring will attempt to join or form a ring. A node cannot belong to more than one ring, and having two “intersecting” rings is highly undesirable and should be avoided. If a node hears an existing ring, the strategy is to wait patiently to join the ring.

If a node is not hearing anything, it is hard to distinguish between the following scenarios:

1. No ring, and potentially other nodes that could form a ring.
2. A large ring with long transit time.
3. An existing ring that is hidden by poor HF conditions

L.5.4. **Ring Maintenance Strategy**

Once a node is in a ring, efforts should be made to maintain the ring. Node failure must be allowed for, but is unlikely. The most common failure is expected to be due to poor HF conditions, and retry is the best approach here. It is also possible that surface wave distance is increasing between a pair of nodes, leading to the link between these nodes becoming not being viable.

L.5.5. **Parameter and Timer Table**

Table L-14 sets out parameters and timers. Some timers specify use of jitter to vary the time to prevent nodes deadlocking with repeat cycle.

Table L-14 – Parameters and Timers

Parameter Name	Default Value	States	Notes
TCLT	TCLT_SILENT	FLT	Claim Token Timer (TCLT) - Timer used in the floating-state (FLT). Controls the time a station waits while in the floating state to claim a token before transiting to another state; a station restarts its TCLT timer when it transits to the FLT state. A node in this state does not know if there are other rings. Primary strategy is to find another ring. If this timer is set too short, risk is that two nodes will form a ring while there is another ring. Set too long, and it will delay setting a ring. There are two values for TCLT, which are set by the state machine. TCLT is initialized to TCLT_SILENT on system start up.
TCLT_SILENT	2 minutes	FLT	This value of TCLT is used when no transmission has been heard recently.
TCLT_TRANS_HEARD	6 minutes	FLT	This value of TCLT is set when a transmission has been heard (that is not message.INVITE) and it can be inferred that a ring exists. In this situation, it makes sense to simply wait for an invitation to be issued. This is likely to take a number of ring cycles and a longer timer is appropriate. It is likely that the ring will continue to be heard before an invite is issued, so this timer will generally get reset.

Parameter Name	Default Value	States	Notes
TSLS	n/a	SFR	Solicit Successor Timer (TSLS) - This timer is used in the self-ring-state (SFR). If it times out the station shall transit to the seeking-state (SEK). The timer shall be set with a random timeout to reduce the probability of collisions by transmissions from stations attempting to establish different rings at the same time. This timer is derived from the following algorithm Random Number in range 0 to NUM_SEK_SLOTS multiplied by SEK_SLOT_TIME
NUM_SEK_SLOTS	3	SFR	Number of asynchronous seeking slots; a parameter to calculate the TSLS timer. Collisions are not expected to be common, so a fairly small number is reasonable. This variation is included to address unusual scenarios where multiple nodes reach FLT at the same time.
SEK_SLOT_TIME	5 seconds	SFR	Time of the slots for TSLS timer. There is no slot co-ordination, so the slot should be somewhat longer than the expected message.INVITE transmission time
TSRP	n/a	SRP	Solicit Reply Timer (TSRP) - Timer used in the solicit-reply-state (SRP). A station starts its solicit-reply timer when it transits to the SRP state. If it expires a station will transit to the JON state where it replies to one of the received SLS tokens, if any, with a SET token. This timer is randomized, as it is possible that multiple nodes will hear message.INVITE and send message.JOIN in response This timer is derived from the following algorithm Random Number in range 0 to NUM_SLS_SLOTS multiplied by SLS_INTERVAL
SLS_INTERVAL	5 seconds	SRP	Time of the slots for TSRP timer. There is no slot co-ordination, so the slot should be somewhat longer than the expected message.INVITE transmission time
NUM_SLS_SLOTS	3	SRP	Number of asynchronous seeking slots; a parameter to calculate the TSRP timer. Collisions are not expected to be common, so a fairly small number is reasonable. The number of slots increases the time in SLT state, which blocks operation of the ring, so this number should not be set higher than needed.

Parameter Name	Default Value	States	Notes
TCON	n/a	JON	Contention Timer (TCON) - Timer used in the joining-state (JON). It controls the time a station waits for a response from another station following an attempt to join the network, so-named because failure to receive a response is attributed to contention with other stations attempting to join the network at the same time; a station restarts its contention timer when it goes to JON state. This needs to be tied to the TSRP timer, as the time to wait is linked to the time which the inviting node will wait before it sends. Then it needs to wait the length of a transmission. The following algorithm is used: (NUM_SLS_SLOTS +1) * SLS_INTERVAL - TSRP This approximates to the time when the inviter will send out a JOIN. There is no issue if FLT is moved to prematurely, as a received token will be handled correctly.
TIDL	n/a	IDL	Idle Timer (TIDL) - Timer used in the idle-state (IDL). It controls the time a station waits for its right-to-transmit before transiting to the floating-state (FLT). The following algorithm is based on max time for token to propagate around the ring. MAX_TX_TIME * RCL() * TIDL_FACTOR
TIDL_FACTOR	1.2	IDL	Factor to adjust TIDL timer for retransmissions.
TSLW	n/a	SLT, SEK	Solicit Wait Timer (TSLW) - This timer is used in the soliciting state (SLT) and the seeking-state (SEK) by stations inviting new ring members. When it expires, the inviting station it will update the Transmit Order List. The following algorithm is used, which reflects the algorithm used is SRP state to transmit message.INVITE: (NUM_SLS_SLOTS +1) * SLS_INTERVAL

Parameter Name	Default Value	States	Notes
TPST	10 seconds	MON	Token Pass Timer (TPST) - Used in the monitoring-state (MON). It controls the time a station waits after passing an RTT (or other) token to another station and failing to hear an implicit acknowledgement before considering the right-to-transmit as lost. It is expected that the this will happen very quickly in normal conditions. If this fails, either the token was not received or the transmission from next node is not being heard. For the former condition a short value is best. It is recommended to choose a fairly conservative value as retransmitting too soon leads to problems.
TEOT	127.5 seconds	HVT	Specifies a time to move to MON state if no EOT-Transmit event occurs.
MAX_TOKEN_PASS	3	MON	Specifies the number retransmissions of a *right-to-transmit* after which a station shall stop trying the current node and to try a different node.
MAX_NET_SIZE	255	HVT	The maximum number of nodes allowed in the ring. If there is a fixed number of nodes in operation, setting this value will save the overhead of issuing invitations when the ring is “complete”. Setting a low limit will improve ring performance, but is unfair to excluded nodes. The recommended default is to use the upper bound, which in practice means no limit.
MIN_INVITE_INTERVAL	15 minutes	HVT	Minimum interval between which a node will issues invites. Setting this value high will improve performance. Setting it low will enable new nodes to join more quickly.
MIN_INVITE_CYCLES	4	HVT	Minimum number of complete stable ring cycles between a node issuing invites. Setting this value high will improve performance. Setting it low will enable new nodes to join more quickly.
EARLY_INVITE_COUNT	2	HVT	If a node has issued this number of invites or less, use EARLY_INVITE_CYCLES
EARLY_INVITE_CYCLES	1	HVT	Minimum number of complete stable ring cycles between a node issuing invites when EARLY_INVITE_COUNT is used
RECEIVE_TABLE_EXPIRY_AGE	n/a	HVT	This sets the time when entries should be removed from the receive table. A node is expected to transmit at least once per ring cycle. This timer is tied to the maximum time for a ring cycle and a factor, which allows for a number of cycles and that cycle time may be much shorter than the maximum. The following algorithm is used MAX_TX_TIME * RCL() * RTEA_FACTOR

Parameter Name	Default Value	States	Notes
RTEA_FACTOR	3	HVT	Variable to control RECEIVE_TABLE_EXPIRY_AGE
MAX_TX_TIME	127.5 seconds	HVT, MON	Maximum Transmit Time. This is a general STANAG 5066 control parameter that can be up to 127.5 seconds. By limiting this time, ring transit times will be reduced and latency reduced. However, for bulk transfers, longer times will significantly improve throughput, particularly where only a small proportion of the nodes are sending data.
NUM_OTHER_RING_HEARD	3	Active Ring	Number of transmissions that must be heard from another ring before closing current ring. Setting this too high will delay ring merging and may lead to two nodes transmitting together. Setting it too low may lead to ring tear down when it is not viable to establish a merged ring.
OTHER_RING_TIME	10 minutes	Active Ring	Period within which NUM_OTHER_RING_HEARD transmissions from another ring must be heard before closing current ring. Considerations for setting this parameter are similar.

L.6. TOKEN AND MESSAGE TRANSMISSION

L.6.1. Transmitting Tokens

Tokens are carried in EOWs. Every D_PDU carries exactly one EOW. A token can be carried in any D_PDU which is being sent to the successor, which will include all WTRP messages being sent. STANAG 5066 may require to use EOWs for other purposes. Subject to this, it is desirable to send multiple copies of the token for reliability.

EOWs **may** be sent using the ACK_PDU or the PADDING D_PDU specified in STANAG 5066 Annex C which is a compact way to transfer an EOW. This will be necessary when no other D_PDUs are being sent to the node receiving the token. It can also enable extra token copies to be sent, which can increase resilience of token transfer.

Where possible, the Tokens should be spread out over the length of the transmission, to minimize the effects of fading and data loss.

Token use of EOWs needs to be balanced with other use of EOWs, such as EOWs used by DTS for data rate selection.

L.6.2. Transmitting Messages

L.4.8 specifies which messages are required to be sent. Messages are small and it may be beneficial to repeat them, particularly if a higher transmission speed is chosen. Where Table messages are critical, it is particularly desirable to duplicate.

Where messages are duplicated, it is best to have them at different parts of the transmission, rather than close together.

All messages are processed after the transmission is received, so there is no particular benefit to placing them in a particular part of the transmission.

L.6.3. Speed and Interleaver Selection

WTRP can be operated at fixed speed or at variable speed. This specification provides a recommended maximum transmission speed for each node. When making a transmission, the speed associated with the “slowest” node needs to be considered as maximum speed for the whole transmission. Where speed is highly variable, this may impact the choice of which D_PDUs to send. Data Rate Selection is controlled by the DTS, which is recommended to take into consideration the factors noted here.

When there is a lot of data to send, it is recommended to use the maximum transmission speed with a long or very long interleaver. This is expected to maximize throughput and reliability.

Messages will often be WTRP messages only, which are generally small. Where there is a smaller amount of data a slower speed and shorter interleaver is likely to give better performance.

When sending message.INVITE to a broadcast address, it is recommended to use a conservatively slow speed. Similarly with message.JOIN. Once communication is established, both ends can use SNR to determine an appropriate operational speed.

L.6.4. Length of Transmission

When a node has the token it can choose how long to transmit for, to a maximum of 127.5 seconds. Long transmissions will optimize throughput, which can give significant improvements due to the time needed to transfer tokens around the ring. However this optimization gives a cost of high latency, particularly for a large ring. This is likely to need tuning with policy across the whole ring. When transmissions speeds are higher, it may be desirable to sacrifice some throughput in order to improve latency.

L.7. CHANGES IN EDITION 4

The overall model and service provided by Annex L is unchanged. The state machine has some small changes. The protocol in Edition 4 is completely different to Edition 3 and interoperability is not a goal. This change was made because of significant problems with the protocol in Edition 3.

ANNEX M

INTENTIONALLY BLANK

ANNEX N GUIDANCE ON ADDRESS MANAGEMENT IN STANAG 5066 NETWORKS (Informative)

N.1 INTRODUCTION

This annex provides guidance for allocation and use of STANAG 5066 addresses in operational systems. It defines a block addressing scheme[1] that can be used for formally planned and ad-hoc operations with a minimal amount of system reconfiguration required.

N.2 MANAGED ADDRESSES

This annex **does not** alter the requirements for size, format, or representation of node addresses defined elsewhere in this STANAG, notably but not limited to the following:

- Annex A– Node ADDRESS Encoding for all [S_]Primitives;
- Annex C– Size of Address [encoding in DPDUs];
- Annex C – Source and Destination Address [encoding in DPDUs];

Addresses in this annex are given in the dotted-decimal format defined in Annex C, and abbreviated generically in the form w.x.y.z .

N.3 NODE ADDRESS-BLOCK ASSIGNMENTS

Only full-length (i.e., 28-bit) S'5066 node addresses **shall** be subject to the assignment recommendations of this Annex.

Regional blocks **shall** be defined by the 4 most-significant bits of the full-length address (i.e., the first element w of the dotted-decimal form w.x.y.z).

[1] This Annex bases its address-block allocation strategy on a proposal put forth by the US Navy as part of their concept of operations for the Battle-Force E-Mail 66 system, which is one of the larger deployments of STANAG-5066 based systems managed under a single operational authority.

Node addresses less than full-length (i..e, with w = 0) **may** be assigned and managed on an ad hoc basis in unique situations; they are otherwise unmanaged by any provision of this Annex. Management authorities for less-than-full-length addresses are not defined.

Managed regional address blocks **shall** be defined by the 4 most-significant bits of the full-length address, in accordance with the Table below.

Table N-1 — Top-Level Address-Block Assignments by Region

Address Range	Regional Assignee	Management POC	Comment
0.0.0.0 — 0.255.255.255	unassigned	ad-hoc	Variable-length addresses are unmanaged
1.0.0.0 — 1.255.255.255	United States	US DoD	includes US Armed Forces and Homeland Security as major S'5066 users
2.0.0.0 —	North America	TBD	
3.0.0.0 —	North America	TBD	other than US government
4.0.0.0 —	South America	TBD	
5.0.0.0 —	NATO	TBD	
6.0.0.0 —	Europe	TBD	
7.0.0.0 —	Europe	TBD	
8.0.0.0 —	Asia	TBD	
9.0.0.0 —	Asia	TBD	
10.0.0.0 —	Africa	TBD	
11.0.0.0 — 11.255.255.255	Middle East/Central Asia	TBD	
12.0.0.0 — 12.255.255.255	Australasia, New Zealand, and Oceana	TBD	
13.0.0.0 — 13.255.255.255	Non-Governmental Organizations	TBD	
14.0.0.0 —	other	TBD	
15.0.0.0 —	other	TBD	

Authority to assign and manage a specific address for a specific system **may** be devolved (perhaps more than once) to a regional or sub-regional administrator. Devolution of management authority within a regional block is beyond the scope of this Annex.

A system's address need not—indeed, should not—be changed when it moves from one geographic region to another. Regional membership and responsibility to assign an address to a system is based on the region in which the system is registered or affiliated (both of these processes are outside of the scope of this standard), rather than the system's location. The major exceptions to this provision are systems belonging to nations involved in NATO operations, where the option of using a national or NATO address may be exercised. Even in this case, however, the election to use an address in one block or another is based on administrative or operational criteria, not physical location.

Whether or not an address manager is assigned for a given administrative region, nodes **should** be assigned addresses from a regional block with which they are registered or affiliated. This will reduce the risk of assigning an address already in use, even absent more active administrative co-ordination.

Entities or individuals wishing to nominate themselves for the role of regional / sub-regional address administrator should contact the NATO C3 Staff (Attn: Network Domain Branch), NATO Headquarters, Brussels, BE.

Within the global-regional blocks, specific 5066 addresses **shall** be assigned to individual nations using the second element "x" of the dotted-decimal address form w.x.y.z. Multiple values of "x" can be pre-assigned in order to give Nx65535 address nodes per nation or organizational unit. Exceptions to this general rule are the US Government and NATO, which will likely be large-scale implementers of the STANAG and are assigned their own unique regional "w" blocks.

Sub-regional blocks allocations are defined in the subsections below for NATO Nations and Partners. Other Nations and organizations may be assigned sub-regional blocks in the appropriate region as their National allocation on a case-by-case basis in coordination with the address administrator identified above.

N.3.1 North-American National Address Schema

Sub-regional assignments for North American National and organizational entities are defined below.

The US Government, as a large scale implementer and deployer of S'5066 systems, is allocated the 1.x.y.z address block and organizational address assignments within this block **shall** conform to the table below.

Table N-2 — North American (US Government) National Addressing Schema (1.x.y.z)

"1" . "x"	Organization		"1" . "x"	Organization
1.1	US Navy		1.8	US Joint Forces Commands
1.2	US Marine Corps		1.9 – 19	Other US Military
1.3	US Air Force		1.20	US Federal Emergency Management Agency
1.4	US Army		1.21	BATF
1.5	US Special Operations Forces		1.22	US Federal Bureau of Investigation
1.6	US Coast Guard		1.23 – 1.255	Other US Government
1.7	US Military Sealift Command			

S'5066 system address allocations for North American NATO Nations and Partners—excluding systems registered with the US Government —**shall** conform to the table below.

Table N-3 — North American (non-US Government) National Addressing Schema (2-3.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
2.1-2.4	(unassigned)		2.8-2.24	(unassigned)
2.5-2.7	Canada		3.1	United States (non-US Government)

N.3.2 South American National Address Schema

S'5066 system address allocations for South American nations **shall** conform to the table below.

Table N-4 — South American National Addressing Schema (4.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
4.1-4.4	(unassigned)		4.6-4.12	(unassigned)
4.5	Colombia			

N.3.3 NATO Address Schema

NATO is expected to be a large user of S'5066 systems and consequently has its own top-level address block assigned to it. Sub-allocations within this address block are made to NATO Nations and commands and **shall** conform to the table below.

Table N-5 — NATO Addressing Schema (5.x.y.z)

"w" .	Country		"w" .	Country
5.1	Belgium		5.12	Norway
5.2	Czech Republi		5.13	Poland
5.3	Denmark		5.14	Portugal
5.4	France		5.15	Spain
5.5	Germany		5.16	Turkey
5.6	Greece		5.17	United Kingdom
5.7	Hungary		5.18	United States
5.8	Iceland		5.19	Allied Command for Operations (ACO)
5.9	Italy		5.20	Allied Command for Transformation (ACT)
5.10	Luxembour		5.21	NATO CIS Group (NCISG)
5.11	Netherland s		5.22	NATO Communications and Information Agency (NCIA)

Allocations within the NATO block are made for national forces provided in support of NATO operations, e.g., to a NATO Response Force (NRF), Combined Joint Task Force (CJTF) or Standing NATO Force (SNF). National forces participating in a NATO mission or operation **should** use addresses in the NATO block, if available. However, nations with NATO block allocation **may** elect to use addresses within their own Nationally allocated block instead of their NATO allocation. Other NATO Nations without an NATO allocation, and non-NATO Nations, **should** use their National allocation. The option to use a nationally administered address or a NATO administered address for any given system should be resolved as part of the operational planning for any mission in which the system participates.

N.3.4 European National Address Schema

S'5066 system address allocations for European Nations **shall** conform to the table below.

Table N-6 — European National Addressing Schema (6-7.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
6.1	Albania		6.27	Lithuania
6.2	(unassigned)		6.28	Luxembourg
6.3	Austria		6.29	North Macedonia
6.4	Belarus		6.30	Malta
6.5	Belgium		6.31	Republic of
6.6	Bosnia and		6.32	(unassigned)
6.7	Bulgaria		6.33	Netherlands
6.8	Croatia		6.34	Norway
6.9	(unassigned)		6.35	Poland
6.10	Czech Republic		6.36	Portugal
6.11	Denmark		6.37	Romania
6.12	Estonia		6.38, 39, 40	Russia
6.13	Finland		6.41	(unassigned)
6.14, 6.15	France		6.42	Slovakia
6.16, 6.17	Germany		6.43	Slovenia
6.18	Greece		6.44	Montenegro
6.19	(unassigned)		6.45	Spain
6.20	Hungary		6.46	Sweden
6.21	Iceland		6.47	Switzerland
6.22	Ireland		6.48	Turkey
6.23, 6.24	Italy		6.49	Ukraine
6.25	Latvia		6.50, 51, 52	United Kingdom
6.26	(unassigned)		6.53	Serbia

N.3.5 Asian National Address Schema

S'5066 system address allocations for Asian Nations **shall** conform to the table below.

Table N-7 — Asian National Addressing Schema (8-9.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
8.1-8.7	(unassigned)		8.11-8.13	(unassigned)
8.8	Japan		8.14	Mongolia
8.9	(unassigned)		8.15-8.22	(unassigned)
8.10	Republic of Korea			

N.3.6 African National Address Schema

S'5066 system address allocations for African nations **shall** conform to the table below.

Table N-8 — African National Addressing Schema (10.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
10.1	Algeria		10.33	(unassigned)
10.2-10.15	(unassigned)		10.34	Morocco
10.16	Egypt		10.35-10.48	(unassigned)
10.17-	(unassigned)		10.49	Tunisia
10.32	Mauritania		10.50-10.52	(unassigned)

N.3.7 Middle East/Central Asia National Address Schema

S'5066 system address allocations for Middle Eastern/Central Asian nations **shall** conform to the table below.

Table N-9 — Middle East National Addressing Schema (11.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
11.1	Afghanistan		11.12	(unassigned)
11.2	Armenia		11.13	(unassigned)
11.3	Azerbaijan		11.14	Pakistan
11.4	Bahrain		11.15	Qatar
11.5	Georgia		11.16	(unassigned)
11.6	(unassigned)		11.17	(unassigned)
11.7	Iraq		11.18	Tajikistan
11.8	Jordan		11.19	Turkmenistan
11.9	Kazakhstan		11.20	United Arab
11.10	Kuwait		11.21	Uzbekistan
11.11	Kyrgyzstan		11.22	(unassigned)

N.3.8 Australia, New Zealand, and Oceania National Address Schema

S'5066 system address allocations for Australia, New Zealand and Nations in Oceania **shall** conform to the table below.

Table N-10 — Australia, New Zealand and Oceana National Addressing Schema (12.x.y.z)

"w" . "x"	Country		"w" . "x"	Country
12.1	Australia		12.7	New Zealand
12.2-12.6	(unassigned)		12.10-12.15	(unassigned)

N.3.9 Non-Governmental Organization (NGO) and Other Address Schema

S'5066 system address allocations for Non-Governmental Organizations and other entities **shall** conform to the table below.

Table N-11 — Non-Governmental Organization and Other Addressing Schema (13-15.x.y.z)

"w" . "x"	NGO		"w" . "x"	NGO
13.1	United Nations		14.0-255	(unassigned)
13.2	International Committee of the Red Cross (ICRC)		15.0-255	(unassigned)

N.4 Changes in Edition 4

Removal of non-NATO, non-NATO Partner Nations from tables and modified allocation procedures accordingly.

Removal of Ether-Client address management, as no longer applicable.

ANNEX O HF Operator Chat (Optional)

HF Operator Chat (HFCHAT) provides a simple and efficient chat communication between operators for exchange of short IA5 text messages. It can also be used to broadcast message to all operators. If provided, the HFCHAT client **shall** conform to the minimal requirements defined herein. The HFCHAT client is intended as a simple mechanism to allow subnetwork operators to test and coordinate their system configurations. It is not a general purpose chat system.

O.1. Changes in This Edition

The text of this annex is taken from Annex F Section F.7 of Edition 3.

The functional differences between this specification and Edition 3 are set out in Section O.4.

There are no significant functional changes.

O.2. General Requirements for HFCHAT

HFCHAT clients **shall** use the ITA5 / ASCII character set to exchange short text messages between subnetwork operators.

HFCHAT messages **shall** consist of sequences of characters terminated by a carriage-return/line-feed pair (i.e., terminated by the octet-pair 0x0D, 0x0A).

The HFCHAT messages length, i.e., the number of octets in the character sequence and including terminating carriage-return/line-feed pair, **shall** not exceed the subnetwork MTU size.

In general, methods of presentation and display to the operator of HFCHAT messages, as well as methods for text-message entry, are beyond the scope of this STANAG and left as implementation options. The following implementation guidelines are recommended, however:

- HFCHAT clients **should** provide a common entry and display area for HFCHAT messages.
- HFCHAT clients **should** provide a short, viewable history of previous messages sent and received (e.g., of the last N messages, N a value (configurable or not) in the range [10,100]).
- HFCHAT clients **should** provide an indication of the source of any HFCHAT messages that it receives (e.g., by displaying the STANAG 5066 address of the originator of the HFCHAT message)
- HFCHAT clients **should** provide an indication of the time-of-receipt of any

HFCHAT messages that it receives.

- HFCHAT clients **should** provide confirmation-of-delivery (node delivery) indications for HFCHAT messages it sends when they are sent using ARQ delivery service.
- HFCHAT clients **should** provide both 1:1 and broadcast transmission.

O.3. Subnetwork Service Requirements

An HFCHAT client **shall** bind to the HF Subnetwork at SAP ID 5.

Priority of messages **should** be configurable and **may** be operator-selectable. Default priority **shall** be 12.

HFCHAT clients **may** use either point-to-point or point-to-multi-point addressing modes to send HFCHAT messages.

The default subnetwork-service requirements when using the point-to-point addressing mode **shall** be as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

For point-to-point operation, the address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the remote HFCHAT client to which the message is addressed.

The default subnetwork-service requirements when using the point-to-multipoint addressing mode **shall** be as follows:

- Transmission Mode = non-ARQ; number of repeats (configurable)
- Delivery Confirmation = NONE
- Deliver in Order = NOT IN-ORDER DELIVERY

For point-to-multipoint operation, the address in the primitive will be a multicast-address corresponding to the group of remote HFCHAT clients to which the message is addressed. Establishment of the multicast address for the group may be done through standard operating procedure or out-of-band channel. Option to use the default broadcast address **should** be provided.

The message's sequence of characters **shall**(1) be bit- and byte-aligned with the octets in an S_Primitive's U_PDU, with the least-significant bit (LSB) of each character aligned with the LSB of the octet. The unused eighth (i.e, MSB) of the octet **shall**(2) be set to zero.

HFCHAT messages **shall** be encapsulated and sent within S_PRIMITIVES, one message per S_PRIMITIVE, with the message-terminating carriage-return/line-feed pair encapsulated in the S_PRIMITIVE as the last two octets of the U_PDU field.

O.4. Changes Since Edition 3

Minor clarifications and corrections made.

ANNEX P ACP 127 and Character-Oriented Serial Stream (Optional)

The Character-Oriented Serial Stream (COSS) is primarily intended for operation of ACP 127 formal messaging over STANAG 5066 ARQ service. COSS is specified as a generic service, so that it cold be used by other applications with similar transport requirements. The ACP 127 protocol transmits a stream of ITA2 or IA5 characters over a serial line. The COSS service provides a service equivalent to serial line transmission. It provides a replacement for ACP 127 operation directly over an HF modem.

P.1. Changes in This Edition

The text of this annex is taken from Annex F Section F.3 of Edition 3.

The functional differences between this specification and Edition 3 are set out in Section P.8.

There are no significant functional changes.

P.2. CHARACTER-ORIENTED SERIAL STREAM (COSS) CLIENT

This annex defines a character-oriented serial-transport service for the HF subnetwork. This provides a reliable emulation of a serial line type service.

The character-oriented serial stream (COSS) service **may** be used in place of other HF serial transport services, for example a simple modem. To provide high reliability and end-to-end assurance of data delivery, the COSS client uses the STANAG 5066 ARQ mode to provide higher data reliability than simple transmission over a conventional modem might afford.

COSS defines two modes of operation:

1. Integrated Operation
2. Baseband Operation

An implementation of this annex **may** offer one or both modes.

P.3. Integrated Operation

COSS may be integrated directly with ACP 127 or another application as shown in Figure P-1.

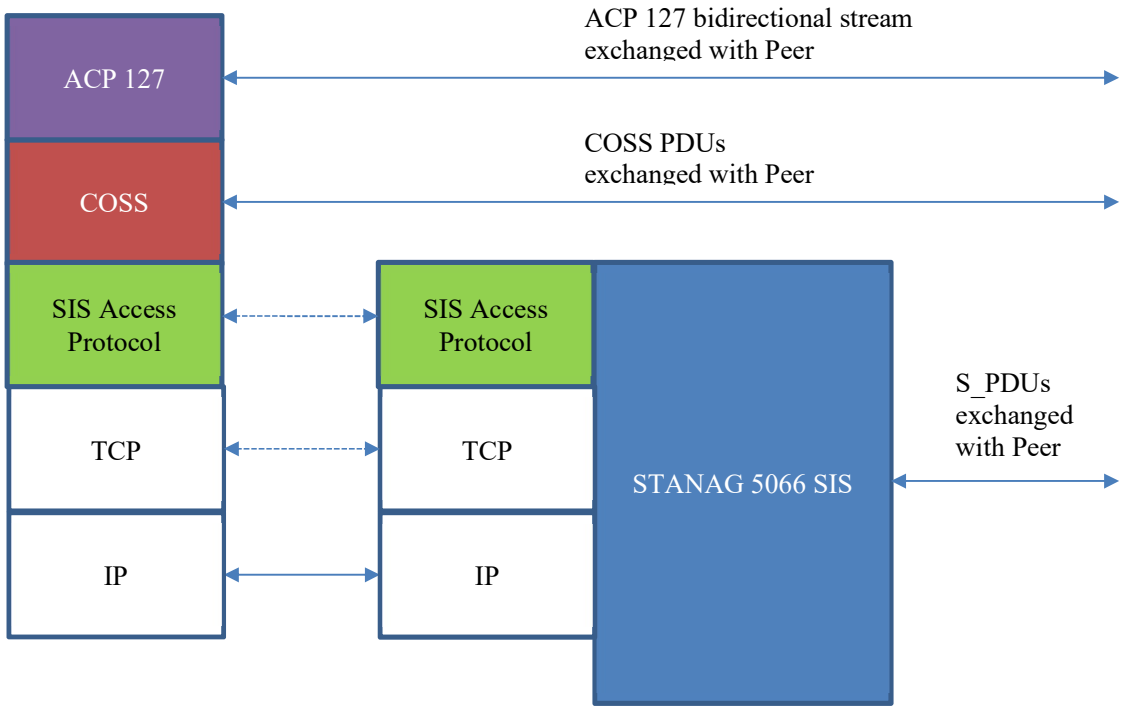


Figure P-1: ACP 127 Integrated Mapping onto COSS

Figure P-1 shows how an application can use a COSS client directly to communicate with a STANAG 5066 server using the SIS Access Protocol. The service provided by the COSS layer enables peer ACP 127 servers to exchange a stream of data in each direction.

P.4. Baseband Operation

COSS can also be configured with a serial interface, so that the COSS module interfaces to ACP 127 or other applications using a serial interface and communicates with a STANAG 5066 server using SIS Access Protocol as specified in Annex S. This mode is termed “baseband operation”, as it can directly replace an application using a baseband serial connection directly to an HF modem.

The interfaces for the Character-Oriented Serial Stream (COSS) Client using Baseband operation are as shown in the Figure below.

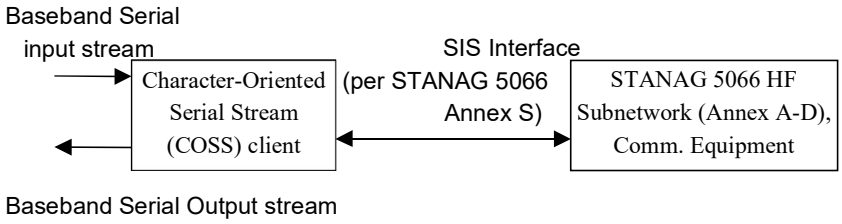


Figure P-2: COSS Client Interfaces

Requirements for the COSS client are placed on its baseband serial interface, its interface to the HF Subnetwork Interface Sublayer (SIS), and its internal processing. Implementations of the COSS client following Baseband operation **shall** use a baseband serial interface, and implementations **shall**(2) be in accordance with the requirements stated herein.

P.4.1. Base-band Serial Interface

A baseband mode COSS client **shall** provide a baseband serial interface meeting the following requirements.

a. *Physical/Electrical*: one of following physical interfaces **shall** be provided:

1. Signalling and connector conforming to EIA/RS-232, EIA//RS-530, configured as Data Communications Equipment (DCE);
2. Signalling and connector conforming to V.35 DCE.

b. *Serial Transmission Mode*: all of the following transmission modes **shall** be supported (as configuration options):

1. Asynchronous: 1 Start Character, selectable 5, 6, 7, or 8 data bits, and 1 or 2 stop bits.
2. Synchronous: provide/accept clock at 1x Data Rate (other rate-multipliers **may** be supported);
   HDLC line control protocol (other synchronous line protocols **may** be supported in addition to HDLC)

c. *Flow-Control*: all of the following flow-control disciplines **shall** be supported (as configuration options):

1. RTS/CTS, DTR/DTS hardware handshaking;
2. XON/XOFF software flow-control;
3. None.

P.5. Character Sets Supported by COSS

The character sets shown in the Table below **shall** be supported by a COSS client:

Table P-1: Character Sets Supported by COSS

Character Set	Comment
ITA-2 / Baudot	5-bit character sets: w/ asynchronous protocol = ITA-2; w/ synchronous protocol =
6-bit codes	6-bit character codes
ITA-5 / ASCII	~ASCII - 7-bit character set [NB: the 7.0-unit 64-ary ITA-2 code defined in STANAG 5030 is encoded in this form, in accordance with section F.3.4.4.1]
Octet data	Any data presented in arbitrary 8-bit formats

P.6. Subnetwork Service Requirements for COSS

COSS clients **shall** bind to the HF Subnetwork at SAP ID 1.

A COSS client **shall** submit its PDUs to the HF subnetwork using the S_UNIDATA_REQUEST Primitives defined in Annex A of this STANAG.

The address in the primitive **shall** be a STANAG 5066 address corresponding to the HF subnetwork address of the host at which the destination COSS client(s) is/are located.

The default service requirements defined when the client binds to the subnetwork **shall** be as follows to support point to point operation:

1. Transmission Mode = ARQ
2. Delivery Confirmation = NONE
3. Deliver in Order = IN-ORDER DELIVERY

COSS **may** also be mapped onto the Non-ARQ transmission mode to support point to multi-point delivery. As ACP 127 is designed to operate over a serial link, Unidata loss from the non-ARQ service may lead to gaps in messages received or to merging

of different messages.

P.7. Data Encapsulation Requirements

The characters from the character stream **shall** be encapsulated within S_PRIMITIVEs using any one of the modes described herein. Implementation of all modes is **mandatory** for a COSS client.

Selection of any given mode for operation **shall** be a configuration parameter in a COSS client, and dependent on the character-set for which the COSS client is configured.

Selection of any given mode **must** be coordinated at the sending and receiving node for use on a given link, through either standard operating procedure or out-of-band coordination channel.

P.7.1. Encapsulation of Arbitrary Octet Data

Octet data in any arbitrary format for the COSS client **shall** be byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE.

The least-significant bit (LSB) of each character received on the serial interface **shall** be aligned with the LSB of the octet.

P.7.2. Encapsulation of ITA-5

Characters in ITA-5 format (or other 7-bit character format such as ASCII) for the COSS client **shall** be aligned with the octets in each U_PDU encapsulated in the S_Primitive, one character per octet, as follows.

	MSB				Octet Data			LSB
Octet / Bit	7	6	5	4	3	2	1	0
$Octet_I$	0	msb	-	-	$ITA5_j$	-	-	lsb

The least-significant bit (LSB) of each 7-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the MSB bit of each octet **shall** be set to zero for ITA5 Encapsulation.

P.7.3. Encapsulation of ITA-2

Two methods of encapsulation of ITA-2 characters are defined:

1. 'Loose-Pack ITA2 Encapsulation' (LPI2E), and
2. 'Dense- Pack ITA2 Encapsulation' (DPI2E).

A COSS client **shall** implement both methods of ITA2 encapsulation.

Selection of either method **must** be coordinated by sending and receiving node

for use on a given link, through either standard operating procedure or out-of-band coordination channel.

P.7.4. **'Loose-Pack Encapsulation of ITA-2 characters**

The Loose-Pack' ITA-2 Encapsulation (LPI2E) algorithm **may** be used for any character set represented as 5-bit symbols. It transports one 5-bit symbol in each octet within the U_PDU field of S_primitives, using the basic packing arrangement defined in the Figure below.

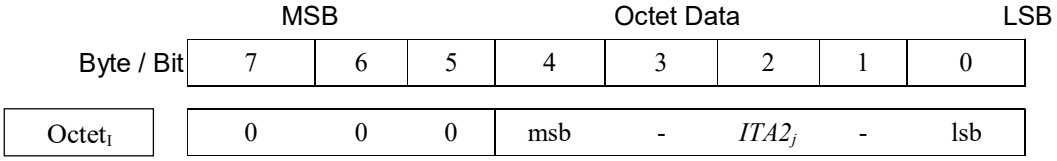


Figure P-3: Loose-Pack Encapsulation of ITA-2 Characters

The least-significant bit (LSB) of each 5-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the three most-significant bits of each octet **shall** be set to zero for LPI2E.

P.7.5. **'Dense-Pack' Encapsulation of ITA-2 characters**

The 'Dense-Pack' ITA-2 Encapsulation (DPI2E) algorithm **may** be used for any character set represented as 5-bit symbols. It efficiently transports three 5-bit symbols in a pair of octets, called an Encapsulation Pair, within the U_PDU field of S_primitives, using the basic packing arrangement defined in the Figure below.

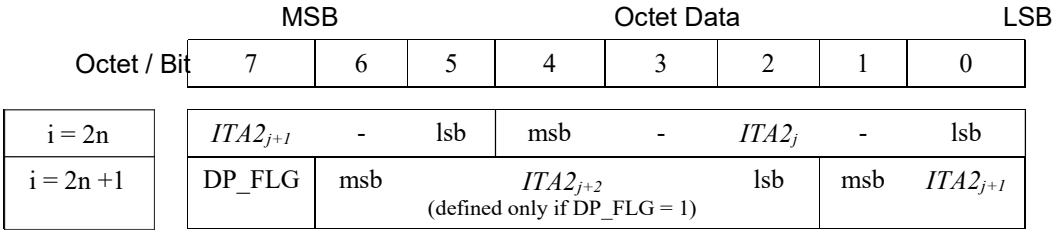


Figure P-4: Nominal Dense-Pack ITA-2 Encapsulation Pair

P.7.5.1.**Encapsulation Pairs**

The first octet of an Encapsulation Pair **shall** be an even-numbered octet in an S_Primitive's U_PDU field (i.e., i = {0,2,4,6, …} , with i = 0 the first octet in the U_PDU).

The second octet of an Encapsulation Pair **shall** be an odd-numbered octet in an S_Primitive's U_PDU field (i.e., i = {1,3,5,7, …} , with i = 0 the first octet in the U_PDU).

The MSB of the second octet of an Encapsulation Pair **shall**(1) be the DP_FLG

('dense-pack flag') that indicates whether the Encapsulation Pair contains three ITA-2 characters (DP_FLG = 1) or two ITA-2 characters (DP_FLG = 0). Encoding of these cases **shall**(2) be performed as follows:

- The first case is defined to be a "Three-into-Two Encapsulation Pair", which **shall** be encoded in accordance with this figure

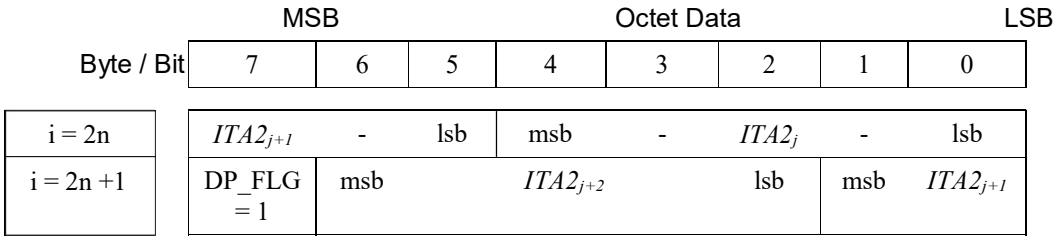


Figure P-5: Three-Into-Two Encapsulation Pair

- the second case is defined to be a "Two-into-Two Encapsulation Pair", which **shall** be encoded in accordance with this figure:

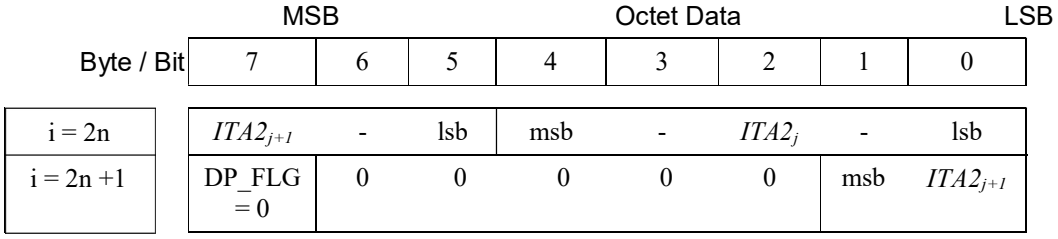


Figure P-6: Two-Into-Two Encapsulation Pair

P.7.5.2. **Maintaining Character-Count Integrity with DPI2E**

The 'Dense-Pack' ITA-2 Encapsulation (DPI2E) algorithm **shall not** add or delete ITA-2 characters from the character stream being transported. This property is denoted character-count integrity.

To maintain character-count integrity, the DPI2E algorithm **shall** depend on the number of characters that remain after all initial characters have been encapsulated as 3-into-2 Encapsulation Pairs.

For a buffer of size B characters encapsulated within a U_PDU field of length L, the DPI2E packing algorithm is defined as follows:

1. **Case R = (B modulo 3) = 0.** In this case, no characters remain after all initial characters are packed in Three-into-Two Encapsulation Pairs:
    - all ITA-2 characters **shall**(1) be densely packed as Three-into-Two Encapsulation Pairs. [i.e.,.: Each Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1].

- The U_PDU length **shall**(2) be set to the value L = (2 * B/3).

2. **Case R = (B modulo 3) = 1.** In this case, one character remains after all initial characters are densely packed in Three-into-Two Encapsulation Pairs:

   - the first (B-1) ITA-2 characters **shall**(1) be densely packed as 3-into-2 Encapsulation Pairs. [i.e.: each of the Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1].
   - The last remaining ITA-2 character **shall**(2) be loosely packed in the last octet of the U_PDU in accordance with the 'Loose-Pack Encapsulation of ITA-2 characters specification' of section F.3.4.3.1.
   - The U_PDU length **shall**(3) be set to the value L = (2 * ((B-1)/3) + 1).

3. **Case R = (B modulo 3) = 2.** In this case, two characters remain after all initial characters are densely packed in Three-into-Two Encapsulation Pairs:

   - the first (B-2) ITA-2 characters **shall**(1) be densely packed as 3-into-2 Encapsulation Pairs. [I.E.: Each Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1]
   - the two remaining ITA-2 characters **shall**(2) be packed as a 2-into-2 Encapsulation Pair. [I.E.: Each Encapsulation Pair will contain two ITA-2 characters, with DP_FLG =0; the bit positions corresponding to the third ITA-2 character in the Encapsulation Pair will be set to zero.].
   - the U_PDU length **shall**(3) be set to the value L = (2 * ((B-2)/3) + 2).

P.7.5.3. Character Unpacking Requirements for DPI2E

In accordance with standard operation procedure or out-of-band coordination circuit, the receiver **must** be configured to perform Dense-Pack ITA-2 Encapsulation.

The receiving client **shall** perform the inverse DPI2E algorithm to unpack the ITA2 characters the U_PDUs contained within an S_UNIDATA_INDICATION primitive:

For a U_PDU of length L,

- If L = 1, the receiving client **shall** unpack the single ITA2 character from the loosely packed octet and send it to the output serial stream;
- If L > 1 and L even, the receiving client **shall**(1) unpack the ITA2 characters in order from each Encapsulation Pair of octets, continuing in order for each successive Encapsulation Pair in the U_PDU. Unpacked ITA2 characters **shall**(2) be sent in the order in which they are unpacked to the

output serial stream. Receiving nodes **may** log a processing error if any Encapsulation Pair except the last has DP_FLG = 0.

- If L > 1 and L odd, the receiving client **shall**(1) unpack the ITA2 characters in order from each Encapsulation Pair of octets, continuing in order for each successive Encapsulation Pair in the U_PDU, and unpacking the last ITA2 character from the single octet (last, and not part of an Encapsulation Pair) in the U_PDU. Unpacked ITA2 characters **shall**(2) be sent in the order in which they are unpacked to the output serial stream. Receiving nodes **may** log a processing error locally if any Encapsulation Pair has DP_FLG = 0. Receiving nodes **may** log a processing error locally if the three most-significant bits of the octet are nonzero.

P.7.6. **Encapsulation of 6-bit Character Codes**

Characters in 6-bit formats for the COSS client **shall** be aligned with the octets in each U_PDU encapsulated in the S_Primitive, one character per octet.

	MSB				Octet Data			LSB
Octet / Bit	7	6	5	4	3	2	1	0
$Octet_I$	0	0	msb	-	$6\_bitChar_j$	-	-	lsb

The least-significant bit (LSB) of each 6-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the MSB bit of each octet **shall** be set to zero for 6-bit character encapsulation.

The special case of the 64-ary ITA-2 character code defined for STANAG 5030 is specified below.

P.7.6.1.**Encapsulation of 64-ary ITA-2 (i.e., STANAG 5030) character codes**

The 64-ary ITA-2 code as defined in STANAG 5030 for Single and Multichannel VLF/LF Broadcasts actually uses a 7-bit character. It consists of 6-bits for information plus a 7^{th} bit (the Stop Bit) that does not change. This allows the possibility that some applications may choose a simple approach to shortening the STANAG 5030 64-ary ITA-2 code to a true six-bit code (i.e., by deleting the stop bit from the code, relying on other measures to provide character synchonization).

In 7-bit format (i.e, unshortened), the 7.0 unit STANAG 5030 64-ary ITA-2 code **shall** be encapsulated as specified in section F.3.4.2.

As a shortened 6-bit code (i.e, with the 7th/stop-bit removed, the STANAG 5030 64-ary ITA-2 code **shall** be encapsulated as specified in section F.3.4.4.

As the overhead from both approaches is equivalent, there is no reason with respect to STANAG 5066 operation to shorten the code. Consequently, use of the unshortened 7.0 unit STANAG 5030 64-ary ITA-2 code is preferred. Other external considerations may apply however that would favor use of the shortened code.

P.7.7. Character-Flush Requirements

It is assumed that the COSS client will be implemented with an input buffer for temporary storage of characters from the stream prior to their encapsulation in S_PRIMTIVES for transmission over the subnetwork. The events that trigger transfer of characters from this buffer to an S_PRIMITIVE (i.e, that triggers a 'character- flush' operation) need to be specified for the client. Of concern are the performance tradeoffs that exist for various character-flush disciplines. For instance, frequent character-flush operations will reduce end-to-end latency and increase the overhead, while for infrequent character-flush operations, triggered only when the size of the input buffer is the subnetwork's Maximum Transmission Unit Size (MTU), the opposite is true. As this is a largely performance issue and not an interoperability issue, a number of different behaviours could be defined.

The COSS client **shall** have a capability to configure the behaviour of its input-buffer character-flush discipline.

Characters **shall** be flushed from the COSS input buffer and encapsulated in an S_PRIMITIVE for transmission over the subnetwork when one or a combination of the following events occur:

1. The number of characters in the input buffer exceeds a configurable threshold value, COSS_BUF_FLUSH_THRESHOLD [NB: if the specified threshold value is greater than the subnetwork MTU size, then COSS_BUF_FLUSH_THRESHOLD shall equal the MTU value.];
2. A carriage-return/line-feed input character-pair is detected in the input character stream. [NB: this behaviour provides line-by-line transmission of a character stream organised as lines of text.]
3. A configurable timeout interval has occurred following the arrival in the input buffer of the last received character. [NB: this behaviour ensures that characters in the input buffer are eventually transmitted if one of the first two events has not occurred.]

Other behaviours for the character-flush disciplines **may** be defined as additional

and configurable implementation options, e.g., triggering a character-flush operation on detection of a user-specifiable character sequence defined as an End-Of-Message (EOM) sequence. [NB: such operation would be useful to support legacy systems such as the ACP-127 messaging systems, which use a defined character sequence as a message delimiter.]

P.8. Changes Since Edition 3

Edition 4 contains a number of minor corrections and clarifications, but is broadly unchanged.

Direct operation of COSS from an ACP 127 implementation without use of serial line is clarified.

ANNEX Q ACP 142 (Optional)

This annex specifies operation of ACP 142 ("P_Mul – A Protocol for Reliable Multicast Messaging in Constrained Bandwidth and Delayed Acknowledgement (EMCON) Environments") over STANAG 5066. This is important for two services.

1. Formal Military Messaging following STANAG 4406, using Tactical Military Messaging protocol specified in STANAG 4406 Annex E.
2. Email using RFC 8494 ("Multicast Email (MULE) over Allied Communications Publication (ACP) 142")

This annex also includes RCOP (Reliable Connection Oriented Protocol) and UDOP (Unreliable Datagram Oriented Protocol) which are used in support of ACP 142 over STANAG 5066.

Q.1. Changes in This Edition

The text of this annex is taken from three sections of Annex F of Edition 3.

- Section F.4 (ACP 142)
- Section F.8 (RCOP)
- Section F.9 (UDOP)

The functional differences between this specification and Edition 3 are set out in Section Q.8.

UDOP and RCOP are included in this annex, as they are used by ACP 142 and wide use of these specifications by other protocols is not anticipated. UDOP and RCOP are unchanged from Edition 3.

Edition 3 Section F.4 was specified for use of STANAG 4406 Annex E, which is based on ACP 142. This annex has generalized this specification so that it is a generic mapping of ACP 142, which did not require any technical changes. It defines use of two protocols over ACP 142

- STANAG 4406 Annex E (TMN-1).
- MULE (RFC 8494) to transfer SMTP.

Q.2. Overall Architecture

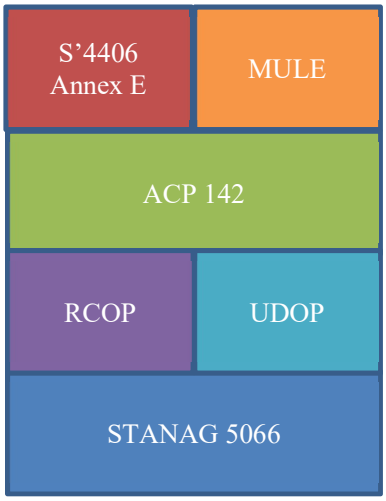


Figure Q-1: ACP 142 over STANAG 5066 Architecture

Figure Q-1 shows the overall architecture for ACP 142 operation over STANAG 5066. ACP 142 has a choice of protocols to interface to STANAG 5066:

1. Reliable Connection-Oriented Protocol (RCOP) is specified in Section Q.6

2. Unreliable Datagram-Oriented Protocol (UDOP) is specified in Section Q.7

ACP 142 is a general purpose multicast transport and can support a range of applications. This annex is focussed on two applications:

1. Formal Military Messaging using STANAG 4406 Annex E over ACP 142. This is described in Section Q.3.

2. Email using RFC 8494 ("Multicast Email (MULE) over Allied Communications Publication (ACP) 142"). This is described in Section Q.4.

Q.3. STANAG 4406 TACTICAL MILITARY MESSAGE HANDLING

A Formal Military Message is different from an interpersonal message in that it is a message sent on behalf of an organization, in the name of that organization, that establishes a legal commitment on the part of that organization under military law, and has been released in accordance with the policies of the originating nation. Examples are military orders. Individuals may send organizational Messages to other individuals on behalf of their respective organizations. Formal Military Messages support the additional services associated with ACP 127, and is therefore interoperable with ACP 127 systems.

Formal Military Messages are handled by Military Message Handling Systems (also

called High Grade Messaging Service). An MMHS takes responsibility for the delivery, formal audit, archiving, numbering, release, emission and distribution of received formal messages on behalf of the originating organization. An MMHS is accountable under military law to provide a reliable, survivable and secure messaging service on behalf of the originating organization. The MMHS fulfills the military messaging service requirements of ACP- 121 and offer high standards of messaging reliability and security. The formal messaging service is seen as the vehicle for secure mission critical, operational military applications.

NATO support for the X.400 protocols in Military Message Handling Systems (MMHS), as defined by NATO STANAG 4406, is mandated in the NATO C3 Technical Architecture (NC3TA), Volume 4 - NATO Common Standards Profile (NCSP). STANAG 4406 Annex E (S'4406E) specifies an adaptation of the X.400/X.500 protocols in STANAG 4406 for Tactical Military Messaging Handling Systems (T-MMHS), with cross-references to STANAG 5066 as one example of a low-bandwidth bearer service to which it has interfaces. This annex summarises and specifies additional requirements for the interface between S'4406E-based T- MMHS and a STANAG 5066-compliant HF subnetwork.

There are several classes of Tactical Messaging Interfaces (i.e., peer-to-peer interfaces between messaging systems) defined by S'4406E, but only one of these (TMI-1) is supported by this annex. TMI-1 is the Tactical Messaging Interface between two Light Message Transfer Agents (LMTA) as specified in S'4406E.

The S'4406E protocol stack defines a compact file format that is passed to ACP 142 for multicast transfer to the recipient peer MTAs. This format and the interface to ACP 142 is fully specified in S'4406E.

Q.4. Multicast Email (MULE)

An equivalent service for standard email is provided by using RFC 8494 ("Multicast Email (MULE) over Allied Communications Publication (ACP) 142"). This specification takes a standard SMTP message and generates a compressed file for multicast transfer by ACP 142. Compression uses the same format as S'4406E. This is fully specified in RFC 8494.

Q.5. ACP 142 Layer

ACP 142 ("P_Mul – A Protocol for Reliable Multicast Messaging in Constrained Bandwidth and Delayed Acknowledgement (EMCON) Environments") provides a reliable multicast transfer service designed to support messaging. It is used by S'4406E and MULE, with service mappings fully defined in those specifications.

ACP 142 is specified to use a datagram service. By default it uses UDP (User Datagram Protocol) specified in RFC 768. The ACP 142 specification clearly notes that any datagram service can be used.

STANAG 4406 Annex E references mapping using Wireless Datagram Protocol (WDP) defined as part of the Wireless Access Protocol (WAP) specification. WDP

defines operation over a variety of transports. As the WAP specification does not include STANAG 5066, the datagram mapping to STANAG 5066 is summarized in STANAG 4406 Annex E Appendix A. This annex specifies additional information on the mapping. If there are any inconsistencies, this annex defines the normative specification.

ACP 142 is mapped onto RCOP and/or UDOP as defined in the following sections. Each ACP 142 PDU is encoded as the User Data of an RCOP or UDOP datagram. RCOP and UDOP traffic **may** be mixed.

Q.5.1. **Mapping ACP 142 onto UDOP**

ACP 142 datagram services make use of the Unreliable Datagram-Oriented Protocol (UDOP) specified in Section Q.7. UDOP **shall** be used for all multicast communication. UDOP **shall** be used for unicast nodes in EMCON state. UDOP **may** be used for unicast nodes not in EMCON state, but this is **not recommended**.

UDOP is based on the STANAG 5066 non-ARQ service, which allows a configurable number of retransmissions. It is **recommended** to not retransmit and to choose transmission speeds where frame loss is anticipated to be low. Data loss is most efficiently handled by ACP 142 retransmission mechanisms and ACP 142 FEC.

It is **recommended** that the ACP 142 MTU size is chosen so that every ACP 142 PDU is transmitted in a single STANAG 5066 UNIDATA. This gives a maximum ACP 142 PDU size of three bytes less than the STANAG 5066 MTU size (typically 2048 bytes). It **may** be desirable to choose a smaller ACP 142 PDU size in order to optimize performance.

Q.5.2. **Mapping ACP 142 onto RCOP**

ACP 142 datagram services **may** make use of the Reliable Connection-Oriented Protocol (RCOP) specified in Section Q.6. RCOP defines a datagram service based on the STANAG 5066 ARQ service. It is **recommended** that RCOP is used for unicast non-ARQ transmission because:

1. STANAG 5066 ARQ retransmission is more efficient, as this is done at D_PDU level, where D_PDU size is generally matched to current HF characteristics. ACP 142 retransmission is a the ACP 142 PDU level; and

2. STANAG 5066 ARQ retransmission is faster, because it is part of the protocol. ACP 142 retransmission is based on a number of timers, which need to be set conservatively to avoid data duplication.

Node Delivery confirmation **shall** be selected. Order **shall** be AS_THEY_ARRIVE.

It is **recommended** that the ACP 142 MTU size is chosen so that every ACP 142 PDU is transmitted in an integral number of maximum size STANAG 5066 UNIDATAs. This will lead to optimal use of STANAG 5066. Use of larger messages will reduce ACP 142 overhead.

RCOP and the underlying STANAG 5066 services generally provide reliable transfer. However, there are error conditions where this does not happen, and the ACP 142 layer **shall** allow for this loss by use of the ACP 142 timers. The operational characteristics over UDOP/Non-ARQ and RCOP/ARQ are very different, so it is **recommended** that timers are independently configurable for each of these modes of operation.

Q.5.3. **Application Identifier**

Both RCOP and UDOP use an Application Identifier to identify the applications used. The following values are assigned by this annex.

Table Q-1: Application Identifiers

Protocol	Hex Value
STANAG 4406 Annex E (TMN-1)	0x2000
MULE (RFC 8494)	0x3000

Q.5.4. **Addressing**

The ACP 142 layer uses its own (four byte) addressing. These addresses need to be mapped onto STANAG 5066 Addresses, as set out in this section.

Each Unicast ACP 142 address **shall** be mapped onto an equivalent STANAG 5066 node address.

The global broadcast ACP 142 address **shall** be mapped onto the default STANAG 5066 broadcast address.

It is **recommended** that ACP 142 dynamic multicast addresses are **not** used with STANAG 5066. If they are used they shall be mapped onto the default STANAG 5066 broadcast address.

For fully interconnected STANAG 5066 networks, there is no benefit to defining static multicast addresses other than a global broadcast address. However, for partially connected WTRP networks, use of static multicast addresses can lead to performance improvements. If this is done ACP 142 static multicast addresses **shall** be mapped onto equivalent STANAG 5066 group addresses.

Q.5.5. **Priority Mapping**

Delivering data according to priority is important in HF systems, particularly for military use. Messages have a priority, which **shall** be used to set priority in the lower layers. Both STANAG 5066 and ACP 142 layers have priorities. ACP 142 priority is 0-255, with 0 as highest priority. ACP 142 can be used for both formal military messaging

and for email, so priority mappings are defined for both services. The priority mapping for formal military messaging is defined in Table Q-2.

Table Q-2: Formal Military Messaging Priority Mapping

Message Priority	ACP 142 Priority	STANAG 5066 Priority
OVERRIDE	3	13
FLASH	5	11
IMMEDIATE	7	9
PRIORITY	9	7
ROUTINE	11	5
DEFERRED	13	3

Email is often operated without priority, but some email services operate with a three-level priority. The three level mapping is defined in Table Q-3. For messages without priority defined, the “normal” mapping **shall** be used.

Table Q-3: Email Priority Mapping

Message Priority	ACP 142 Priority	STANAG 5066 Priority
Urgent	10	6
Normal	12	4
Non-Urgent	14	2

Q.5.6. SAP Selection

SAP assignment is set out in Annex F. There are two ACP 142 assignments, which shall be used for these classes of ACP 142 traffic:

- For Military Formal Messaging using STANAG 4406 Annex E or MULE, SAP 2 **shall** be used.
- For Email using MULE, SAP 7 **shall** be used.

Q.6. RELIABLE CONNECTION-ORIENTED PROTOCOL

This section specifies a simple Reliable Connection-Oriented Protocol (RCOP) for reliable data connections between applications using the ARQ services of the HF subnetwork. RCOP provides a minimal header to support multiplexed connections between a pair of nodes through a single hard or soft link. Applications using an

RCOP connection are identified uniquely by a field in the header of the RCOP PDU.

Q.6.1. **RCOP Protocol Data Unit (RCOP_PDU)**

The format of all RCOP U_PDUs **shall** be as shown in the Figure below.

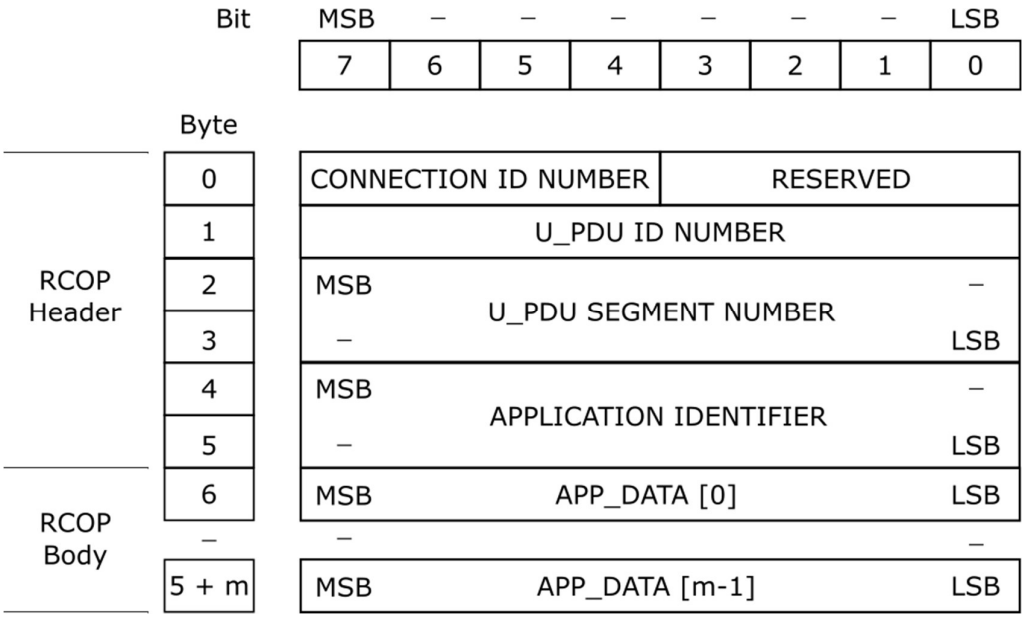


Figure Q-2: Format for Reliable Connection-Oriented Protocol Data Units

The following are required for RCOP PDUs:

1. RCOP clients may provide multiplexed transport service for more than one application simultaneously by establishing multiple connections, each identified by its CONNECTION_ID_NUMBER field. The CONNECTION_ID_NUMBER field **shall** be a value from 0-15.
2. New values for the CONNECTION_ID_NUMBER field **shall** be dynamically assigned as new connections are established. Further details regarding assignment and co-ordination of connection ID numbers are not specified here.
3. Connection ID number 0 **shall** be reserved for non-multiplexed connections or the first multiplexed connection.
4. The reserved bits **shall** be set to 0.
5. U_PDU ID numbers **shall** be assigned consecutively to user PDUs (U_PDUs) serviced by the connection.

6. The U_PDU segment number **shall**(1) be assigned consecutively to segments within a single U_PDU. The first segment transmitted **shall**(2) be assigned segment number 0. If a U_PDU is not segmented, the single segment that is transmitted **shall**(3) be assigned number 0.
7. The APPLICATION_IDENTIFIER **shall** be assigned in accordance with the requirements of section F.10 of this Annex. This field serves to identify the application (i.e., higher-level protocol) using the connection. End-to-end interoperability can be achieved only if this is a unique value for any given end- user application, otherwise there is the (likely) chance that the applications connected by the RCOP client and subnetwork will be incompatible.
8. The APP_DATA[] field **shall** contain the m-bytes of application data sent over the connection. Segmentation and reassembly rules for mapping the application's data into the APP_DATA[] field are in general outside of the scope of this STANAG, and are application dependent.

Q.6.2. RCOP Subnetwork Service Requirements

RCOP clients **shall** bind to the HF Subnetwork at SAP ID 6. Client priority is a configuration- dependent and implementation-dependent parameter.

Each RCOP_PDU sent over the subnetwork **shall** be embedded in an S_UNIDATA_REQUEST primitive, each byte of the RCOP_PDU corresponding to a byte in the U_PDU field of the S_Primitive (see Figure). RCOP_PDUs will be encapsulated in like manner when delivered by the subnetwork to the destination client in an S_UNIDATA_INDICATION.

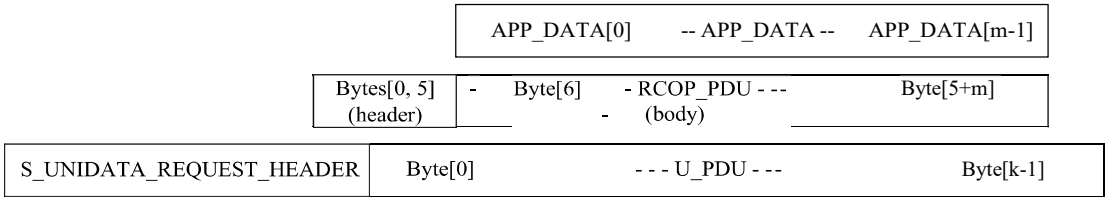


Figure Q-3: Mapping of RCOP_PDU into an S_UNIDATA_REQUEST Primitive for transmission.

The encoded data for the RCOP client **shall** be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

Note that, in accordance with the provisions of Annex A of this STANAG, if the

subnetwork interface sublayer receives a S_UNIDATA_REQUEST primitive with an RCOP Protocol Data Unit larger than the maximum MTU size, the S_UNIDATA_REQUEST will be rejected.

An RCOP client **shall** set the default service requirements for S_UNIDATA_REQUEST primitives as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY or AS_THEY_ARRIVE

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host at which the destination RCOP client is located.

Q.6.3. **RCOP Segmentation and Reassembly Requirements**

Note that, in accordance with the provisions of Annex A of this STANAG, if the subnetwork interface sublayer receives a S_UNIDATA_REQUEST primitive with an RCOP Protocol Data Unit larger than the maximum MTU size, the S_UNIDATA_REQUEST will be rejected. RCOP clients **must** segment their data and place it into the APP_DATA[] field of the PDU accordingly.

If IN_ORDER delivery-order is specified, an RCOP client **shall** simply take the U_PDU segments in the sequence in which they arrive to construct a copy of the original U_PDU sent by the source. If AS_THEY_ARRIVE delivery-order is specified, an RCOP client **shall** be responsible for reassembling the U_PDU segments it receives in proper sequence order.

RCOP clients **may** devise any algorithm of their own choice for segmentation and reassembly, but the RCOP_PDU fields for CONNECTION_ID_NUMBER, U_PDU_ID NUMBER, U_PDU SEGMENT NUMBER are available for such use. Segmentation and reassembly algorithms **shall** not use the APPLICATION IDENTIFIER field.

Note that, in general, a local RCOP client could be receiving data on two different connections, each established by another remote RCOP client as the remote client's sole connection. In this case, the remote clients would each have specified a connection ID number of zero. As an alternate but similar scenario, two RCOP clients on the same remote node but attached to different SAP IDs could connect to the local RCOP client. In this case also, data could be received with the same connection ID number. Other scenarios in which the same connection ID has been assigned to data received by an RCOP client might also occur, as dynamic connections are made and broken with different nodes. Thus, to reassemble segmented application data without ambiguity, an RCOP client **must** distinguish U_PDU segments for its receive connections using the unique combination of (SOURCE_ADDRESS,

SOURCE_SAP_ID, CONNECTION_ID_NUMBER). The SOURCE_ADDRESS is the address of the originator of the RCOP_PDU, and the SOURCE_SAPID is the SAP_ID to which the remote RCOP client is attached. All three parameters the can be obtained unambiguously from the S_UNIDATA_INDICATION primitive in which the RCOP_PDU is delivered to the client, as can the CONNECTION_ID_NUMBER, U_PDU_ID NUMBER, U_PDU SEGMENT NUMBER.

Q.7. UNRELIABLE DATAGRAM-ORIENTED PROTOCOL (UDOP)

This section defines a simple Unreliable Datagram-Oriented Protocol (UDOP) using the non-ARQ services of the HF subnetwork, with a minimal header to support multiplexed datagram delivery. Since non-ARQ services are used, the UDOP may support a multicast service through the use of group addresses within the HF Subnetwork. Applications using a UDOP non-ARQ delivery service are identified uniquely by a field in the header of the UDOP PDU.

Q.7.1. **UDOP Data Unit**

UDOP Protocol Data Units **shall** be defined and used identically to those defined for the Reliable Connection- Oriented Protocol in Section Q.6.1.

Q.7.2. **UDOP Subnetwork Service Requirements**

UDOP clients **shall** bind to the HF Subnetwork at SAP ID 7. Client priority is configuration- dependent and implementation-dependent parameters for this application.
A UDOP client **shall** submit its U_PDUs to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = non-ARQ
- Delivery Confirmation = none
- Deliver in Order = AS_THEY_ARRIVE

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host on which the destination message stores are located. The UDOP client at the destination **shall** be responsible for reassembling the U_PDU segments in proper sequence order.

The encoded data for the UDOP client **shall** be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

Q.8. Changes in Edition 4

ACP 142 is moved from Annex F to this Annex.

RCOP and UDOP are included in this annex, as they are used by ACP 142 and use by other protocols is not anticipated.

There are no protocol changes to the protocols specified here.

The Annex is formulated as ACP 142, rather than STANAG 4406 Annex E. Technically, this does not lead to a change, but it means that the annex now encompasses other protocols using ACP 142, in particular MULE (RFC 8494).

Only the TM-1 (MTA to MTA) option of STANAG 4406 Annex E is included. Other options are not included, as they are not being used.

Priority mappings for the layers are specified.

The Reserved Service Identifiers needed are specified here. They were previously specified in Annex F section F.10.

ANNEX R ROUTING SUBLAYER (Optional)

The Routing Sublayer is an optional sublayer above Channel Access Sublayer (CAS) and below Subnet Interface Sublayer (SIS). The Routing Sublayer supports transfers of data that needs to traverse one subnet twice or needs to traverse multiple subnets. It also supports systems with multiple independent subnets. The primary goal of Routing Sublayer is to support operation over Wireless Ring Token Protocol specified in STANAG 5066 Annex L when there is partial connectivity.

R.1. Routing Sublayer Overview

The terms Subnet is used for a set of nodes connected to an HF (or other) channel communicating using STANAG 5066. SIS (Subnet Interface Service) was originally conceived as a service to access a single subnet/channel. The Routing Sublayer does not change this service, but enables the SIS service to be provided across a concatenation of Subnets.

The STANAG 5066 service defines operation over a single Subnet. This specification adds a routing layer so that the STANAG 5066 SIS service can be provided across multiple Subnets and support repeated transmission over a single Subnet. The repeated transmission is needed to provide data communication between all nodes when Wireless Ring Token Protocol is used with partially connected topology. Routing Sublayer also enables a SIS client to communicate with peers on different Subnets, with the choice of Subnet transparent to the client. Finally, the Routing sublayer enables operation over a set of concatenated Subnets.

R.2. Goals of the Routing Sublayer

R.2.1. **Wireless Token Ring Protocol Support**

Wireless Token Ring Protocol (WRTP) as specified in STANAG 5066 Annex L, defines a framework for supporting communication between nodes sharing a channel where the nodes cannot communicate directly. It specifies a MAC level service that provides information on node connectivity to the higher layers of STANAG 5066.
The routing sublayer defined in this specification makes use of this connectivity information to enable users of the STANAG 5066 service to communicate data to nodes that are not directly connected.

R.2.2. **Routing across Multiple Subnets**

STANAG 5066 defines a service with access to a single channel (typically accessed using a single modem) and service to nodes connected by that channel. A system may be connected to multiple channels with independent STANAG 5066 services. This routing sublayer enables transparent interconnection of such services. Examples as to where this might be useful.

1. A ship with two HF connections: Skywave reach back to shore; WTRP surface wave communication to other ships. This routing protocol would enable them to interconnect.
2. A shore station with skywave access to several mobile units, each with a dedicated channel and STANAG 5066 service. The routing sublayer would enable STANAG 5066 communication between the mobile units.
3. A MANET (Mobile Ad hoc Network), with a mix of HF links and line of sight links (e.g., UHF).

R.2.3. **Transparent Client Access to multiple Subnets**

Where a system has multiple STANAG 5066 Subnets, without Routing Sublayer, there would be a need for each Subnet to have its own STANAG 5066 server. An application using STANAG 5066 would need to connect to the "right" subnet for a given peer. Operating with the Routing Sublayer enables a SIS client to address STANAG 5066 nodes on any connected Subnet through a single SIS interface.

R.3. Alternate Approaches

There are two "obvious" alternatives to the routing sublayer defined here, which are discussed below. Both are highly desirable approaches for some deployments, but they cannot address all scenarios.

R.3.1. **Application Relay**

When transferring application data over HF, it is highly desirable to proceed one hop at a time. This enables communication to be tuned for one network, without introducing relay. This can be done, and is recommended, whenever routing configuration is stable.

However, where routing can change, for example with WTRP, application relay is not viable, because an application relay connecting over SIS will not have access to the dynamically changing WTRP configuration.

Application Relay is useful for some applications such as messaging and XMPP, but other applications such as Web Browsing do not usefully support application relay.

R.3.2. **IP Relay**

Use of IP is a common approach used to build MANETs over multiple subnetworks. In many cases, this is a good and flexible choice. However, HF subnetworks have high and variable latency, which leads to problems deploying these applications over HF. Use of relay over multiple networks would compound these problems. This IP approach will not be generally viable for HF, particularly "bulk" applications using TCP. Research supporting this assertion is provided in the Isode White Paper, Measuring and Analysing STANAG 5066 F.12 IP Client[1]. Another Isode White Paper, Measuring and Analysing HF-PEP for TCP communication and Web Browsing over HF[2] shows

[1] https://www.isode.com/whitepapers/measuring-stanag5066-f12-ip-client.html
[2] https://www.isode.com/whitepapers/hf-pep-measurements.html

that an approach operating directly over STANAG 5066 can address these issues. This solution can provide good performance using Routing Sublayer.

R.4. Routing Layer Architecture

This section provides a recap of the STANAG 5066 Ed3 architecture, and shows how the Routing Sublayer (RS) extends this architecture.

R.4.1. **STANAG 5066 Ed3**

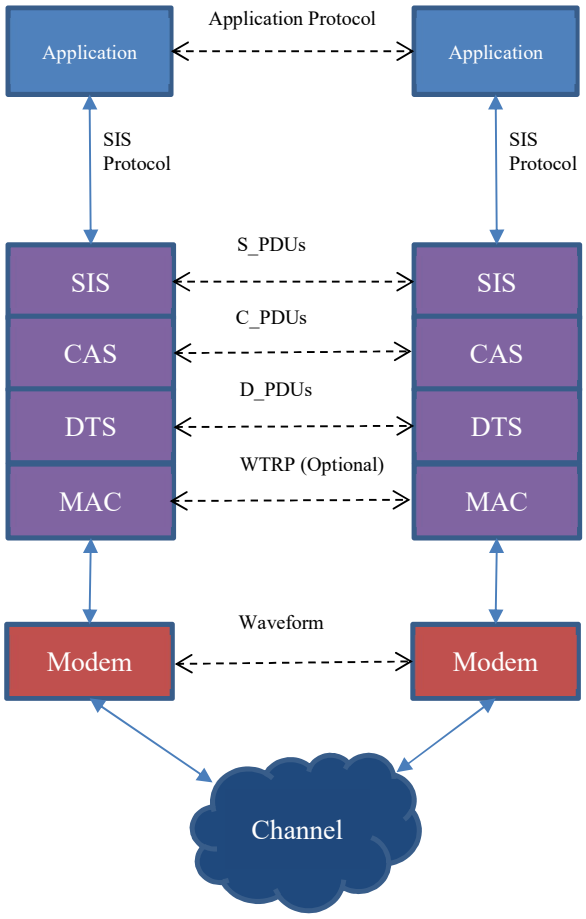


Figure R-1 STANAG 5066 Ed3 Architecture

The STANAG 5066 Ed3 protocol architecture is illustrated above. Notes on this architecture:

1. Modems communicate with a waveform, such as STANAG 5069. There will be other hardware needed to connect a modem to an HF Channel (or to another channel such as UHF).
2. STANAG 5066 Applications communicate with STANAG 5066 servers using the SIS protocol.
3. Applications running over STANAG 5066 define their own end to end application protocols, such as COSS used for ACP 127 messaging.
4. STANAG 5066 is defined as four layers, with PDUs defined at each layer that (except for MAC) are transferred by the protocol layer below. These layers are:
    a. Subnet Interface Sublayer (SIS) which communicates S_PDUs.
    b. Channel Access Sublayer (CAS) which communicates C_PDUs.
    c. Data Transfer Sublayer (DTS) which communicates D_PDUs.
    d. Media Access Control (MAC) sublayer. The communication depends on the MAC layer choice:
        i. The MAC layer is optional and is not required on a channel with just two nodes.
        ii. Annex K (CSMA). The MAC layer is procedure only, and no protocol is exchanged.
        iii. WTRP. This exchanges protocol, but uses special D_PDUs (rather than having MAC layer PDUs).
        iv. TDMA. There is a placeholder (Annex M) for this, but there is no specification.

R.4.2. **Routing with Multiple Independent Subnets**

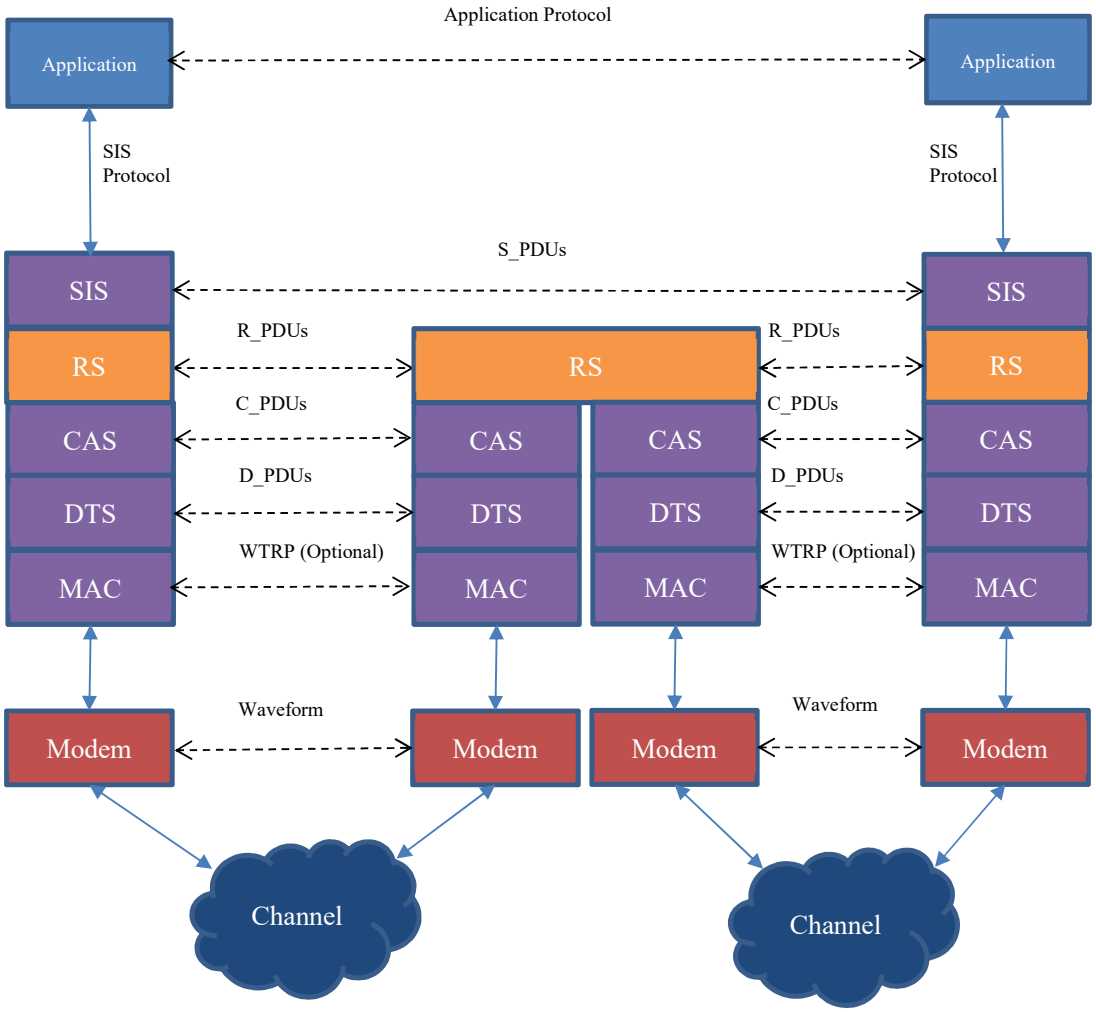


Figure R-2 Routing with Multiple Independent Subnets

Routing Sublayer (RS) is added as a STANAG 5066 layer between SIS and CAS, with exchange of R_PDUs. The SIS protocol is end to end, whereas CAS and the layers below are single hop.

The above diagram illustrates how the routing sublayer is used with two independent channels. If the LHS node in the above diagram sends data to the RHS node which is not connected, the routing sublayer communicates to the intermediate node the destination address in the Routing Sublayer protocol. This enables the middle node to route traffic between the two nodes. Note that this routing is completely transparent to the application.

NOTE: Many STANAG 5066 servers have been developed as modular products, and it may be desirable to provide the routing layer as a separate module, so that it can make use of multiple modular STANAG 5066 servers. It would be clean to use SIS for this purpose, but unfortunately SIS does not provide sufficient information and it is desirable to orient SIS to client access. The separation described here could be achieved by a new protocol to access the CAS service, which might be vendor-specific or standardized as a part of a future edition of STANAG 5066.

R.4.3. **Routing with Single Channel**

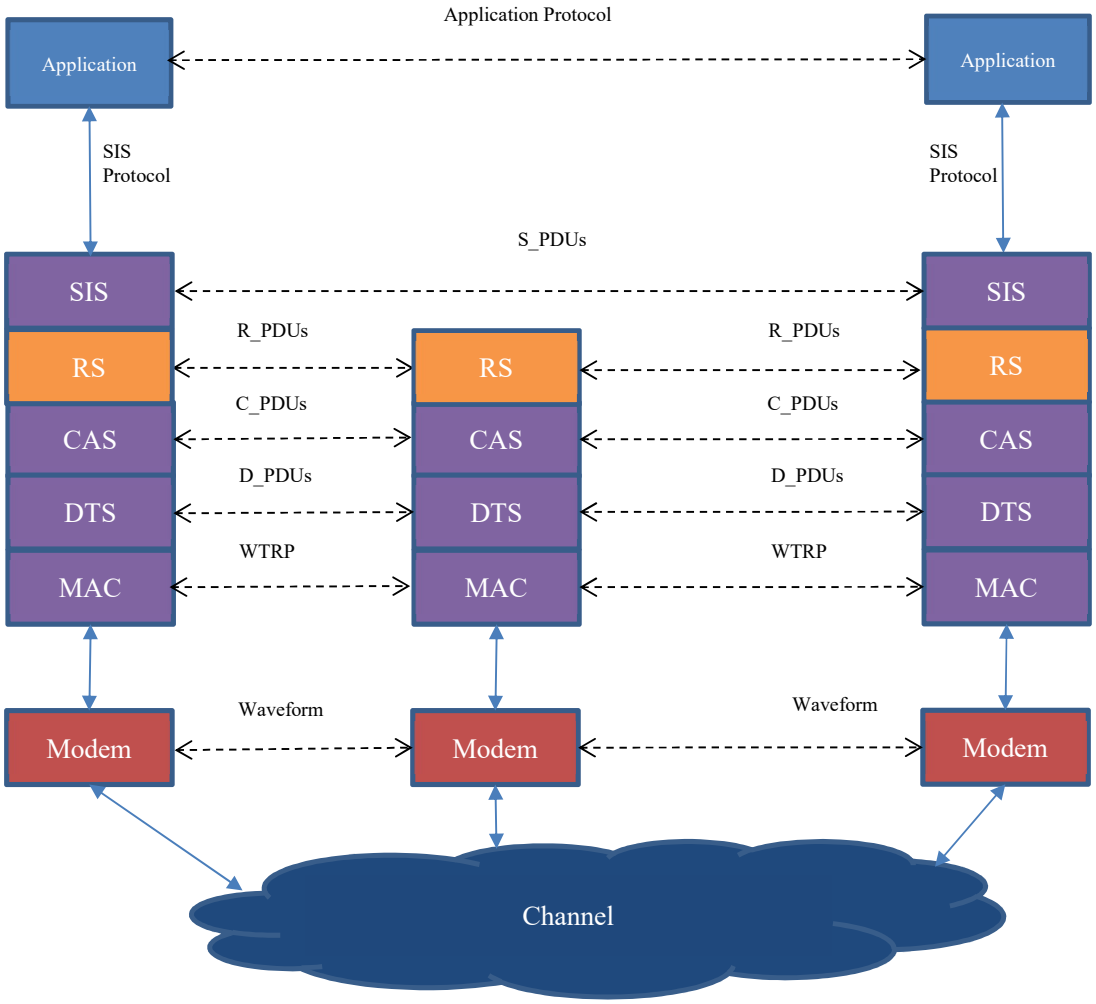


Figure R-3 – Routing with Single Channel

The above diagram shows how the routing sublayer works with a shared channel. WTRP is the only MAC layer that supports such a channel, so this description is given for WTRP, although the routing sublayer would work for a different MAC layer with the same characteristics.

Although all three modems are connected to the same channel, the diagram above relates to a scenario where the middle node can communicate with both of the end nodes, but traffic does not propagate between the end nodes. WTRP supports this configuration and will ensure that only one node transmits onto the channel at any one time.

All transmissions onto the channel are broadcast at the physical layer. When the LHS node transmits data to the RHS node, the address of the RHS node is included in the routing sublayer protocol, and the lower layers send traffic to the middle node. The middle node receives this message which is handled by the routing sublayer, which will send the data through the same stack but addressed to the RHS node. Both RHS and LHS nodes will hear data transmitted by the middle node, but will only handle traffic addressed to them.

The information to control this routing is provided by the WTRP layer specified in Annex L. This defines in Section L.2.9.2 the *next hop table*, which contains the necessary information.

R.4.4. **Transparent Client Access to Multiple Subnets**

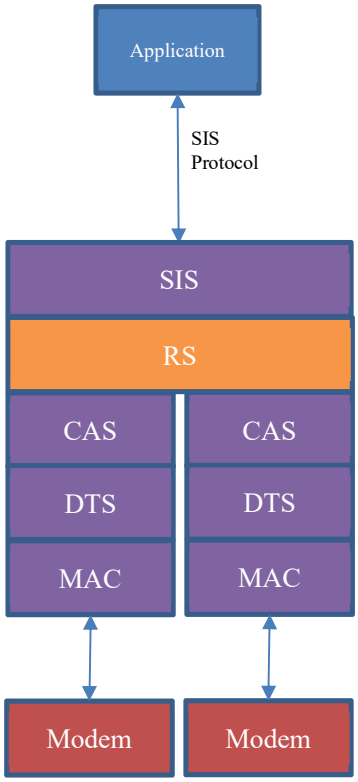


Figure R-4 – Client Access to Multiple Subnets

When a system is connected to two or more modems through independent STANAG 5066 stacks, the routing sublayer (RS) enables SIS provision to an application to transparently access the lower layers based on STANAG 5066 address of the destination. The application does not need to be aware of which modem and stack is used.

R.5. Routing Layer PDUs

The routing sublayer uses five R_PDUs. These use a single leading byte to identify the type of the PDU.

R.5.1. **Direct R_PDU**

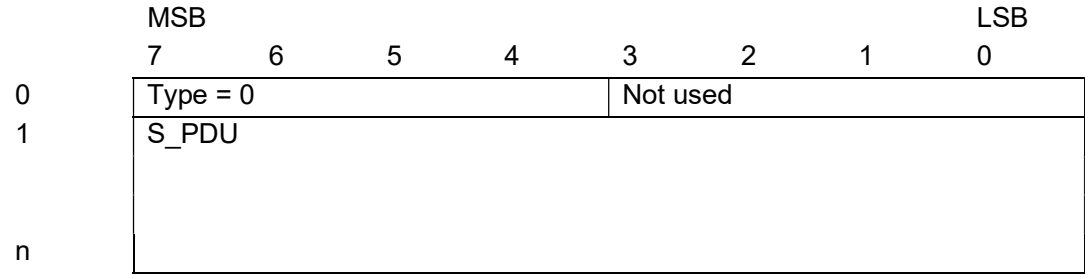
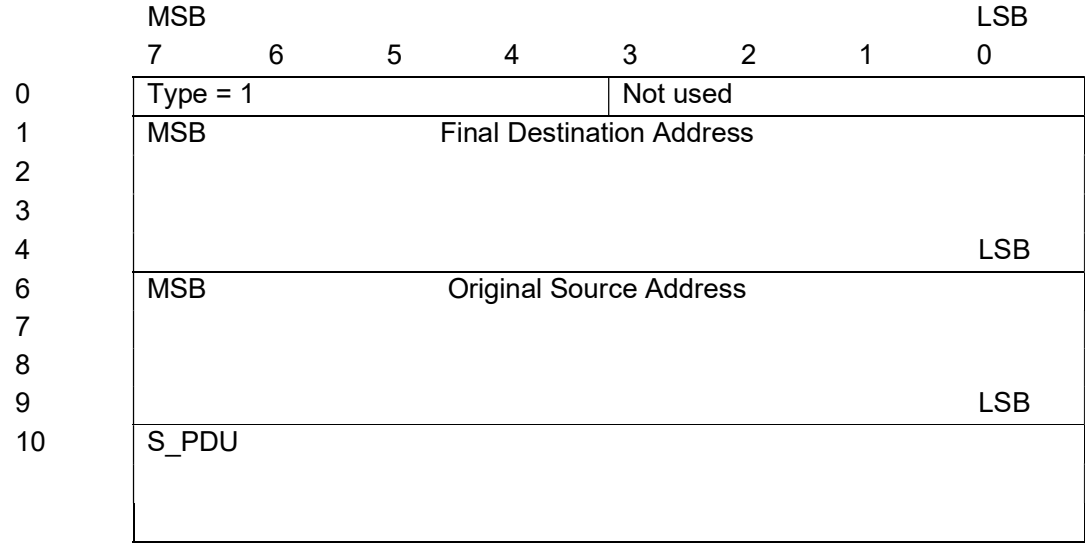


Figure R-5 – Direct R_PDU

The Direct R_PDU is used when the next hop is the final destination for the data. There is a single byte header of Type=0, and the rest of the data is the encapsulated S_PDU.

This R_PDU is for cases where the Routing Sublayer is not needed. It is a minimal PDU that needs to be present to provide a coherent layer service.

R.5.2. **Indirect R_PDU**



n	

Figure R-6 Indirect R_PDU

The Indirect R_PDU is used when the next hop is not the final destination. The Indirect R_PDU is type 1, and includes the STANAG 5066 address of the destination. This will enable CAS and layers below to send the S_PDU to the next hop. It also includes the original source address, to enable the final SIS service to report this address to the recipient.

R.5.3. General Broadcast R_PDU

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type = 2				Not used			
1	S_PDU							
n								

Figure R-7 – Broadcast R_PDU

The General Broadcast R_PDU is used when data is being broadcast (or multicast), to all the nodes on a single subnet This is always used for an initial broadcast/multicast transmission. There is a single byte header of Type=2, and the rest of the data is the encapsulated S_PDU.

R.5.4. **Selective Broadcast R_PDU**

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type = 3				Not used			
1	MSB			Original Sender				
2								
3								
4								LSB
5	MSB			Destination Count=d				LSB
6	MSB							
				Destination Address				
								LSB
9								
10	MSB							
				Destination Address				
								LSB
13								
5 + 4*d	S_PDU							
n								

Figure R-8 – Multicast R_PDU

The Selective Broadcast R_PDU is used when a broadcast/multicast is intended for a specific set or nodes only. This is used when a broadcast/multicast message being relayed or for initial broadcast/multicast when there are multiple subnets connected. The STANAG 5066 address of the node that originally sent the broadcast is included, as this will not be available at on relay. Then there is a list of nodes to which the message needs to be sent. This is to ensure that a message is only re-broadcast when needed.

The number of destination STANAG 5066 addresses is encoded as Destination Count. This allows the list of destination addresses to be determined.

R.5.5. **Routing Update R_PDU**

	MSB 7	6	5	4	3	2	1	LSB 0
0	Type = 4				Not used			
1	Routing update information							
n								

Figure R-9 – Routing Update R_PDU

The Routing Update R_PDU is used to share routing information between nodes on different subnetworks. This is type 2. Routing update is a sequence of five byte information blocks. The Routing Update R_PDU is transmitted without any additional data, so the lengths of this PDU and number of blocks can be determined from the lower layers. Routing updates are only sent directly (single hop) so that the source of the Routing Update can be determined from the lower layers.

Each five byte block is encoded as follows.

	MSB 7	6	5	4	3	2	1	LSB 0
0	MSB			Reported Address				
1								
2								
3								LSB
4	Hop Count							

Figure R-10 – Block Encoding of Routing Update R_PDU

The first four bytes of each routing update information block is the STANAG 5066 address of the node being reported on. The fifth byte is a Hop Count, which indicates the number of routing hops to this node from the node that is sending the routing update. Where the node is directly reachable from the reporting node, hop count is set to 1.

R.6. Procedures of Operation

This section describes how the routing sublayer works.

R.6.1. **CAS Layer Interface**

The interface to CAS layer is modified as follows:

1. Where STANAG 5066 Annex B uses S_PDU, R_PDU is used when the routing sublayer is used.
2. Other parameters to CAS layer are specified in the following procedures.

R.6.2. **Determining Routing**

The routing sublayer needs to know the full set of addressable STANAG 5066 nodes and for each node it has to determine the next hop. This information can be obtained by four mechanisms:

1. Configuration. For example when a subnet is set up with a specified set of STANAG 5066 nodes. This could be a fixed configuration or a configuration updated dynamically by an external process.
2. From Wireless Token Ring Protocol, as specified in Annex L, which supports topologies where nodes are not directly connected. Information on this topology is passed upwards using an Annex J M_ service primitive. This is specified in Annex L as the next hop table which gives a list of all nodes in the ring, and indicates which nodes are directly connected. For nodes that are not directly connected it indicates a preferred data relay node, which is the directly connected node that is recommended to use for data relay. Annex L makes the next hop table available to the routing sublayer.
3. By discovery. There is no standardized mechanism to achieve this. Isode has defined an open protocol to achieve this in “HF Discovery, Ping and Traffic Load” (S5066-APP2).
4. Using the Routing Update procedures of this specification, which enables discovery of nodes which can be routed to indirectly over subnets that are connected indirectly.

NOTE: When used with WTRP, Annex L will determine and provide next hop information based on link quality, which will dynamically adapt to provide the best route. This information does not need to be shared with the routing protocol. The routing mechanism for concatenated networks is based on simple connectivity. It may be desirable, in a future version of this protocol, to enhance routing update to provide link quality information, which can give better routing in complex networks.

R.6.3. **Submission Procedure**

The SIS layer will provide an S_PDU and other parameters to be passed down to the CAS layer.

1. If the target address is a broadcast address, there are three distinct scenarios:
    a. If the S_PDU is provided by the SIS layer and there is a single subnet, encapsulate the S_PDU in a General Broadcast R_PDU. Then pass the R_PDU and submission parameters to CAS layer.
    b. If the S_PDU is provided by the SIS layer and there are multiple subnets for each of the connected subnets determine the list of destination

addresses for each subnet, so that all target addresses are reached by exactly on subnet. Then encapsulate the S_PDU in a Selective Broadcast S_PDU, with the Original Sender set to the local node address and the destination addresses included.

c. If the S_PDU is provided by relay (i.e., when it has been delivered from CAS layer and routing has determined the need for relay), there will be an Original Sender and a list of destination addresses to handle. Determine which subnet will handle each address, and for each subnet submit a Selective Broadcast R_PDU with the set of addresses determined to be on that subnet

2. If the target address is a single node for which the local node does not have routing information, the SIS request is rejected with the C_UNIDATA_REJECT owith reason “Address Not Routable”.

3. If the target address is a single node which can be directly reached on a connected subnet, then encapsulate the S_PDU in a Direct R_PDU and pass to CAS layer with the provided parameters.

4. If the target address is a single node which can be reached indirectly, then encapsulate the S_PDU in an Indirect R_PDU, with the final destination address set to the target address and original source address set to the source address. Then pass to the CAS layer the R_PDU with SIS submission parameters with the target address replaced by the next hop address.

NOTE: because relayed PDUs will inherently have higher latency, it **may** be desirable to transmit them ahead of non-relayed PDUs.

R.6.4. **Reception Procedure**

The CAS layer will provide received R_PDUs to the routing sublayer .Handling of Routing Update R_PDUs is described in the next section. The different PDUs are handled as follows:

1. The S_PDU from Direct R_PDUs are passed directly to the SIS layer.

2. When an Indirect R_PDU is received, the final destination is examined. If it is local, the S_PDU and associated parameters are passed up to SIS layer, with final destination address and original sender address (so that relay is transparent to the receiver). Otherwise, the S_PDU is extracted. Then Submission Procedure is followed using the S_PDU, service parameters associated with the received R_PDU and the target address being the destination address from the Indirect R_PDU. This process is termed relaying.

3. When a Broadcast R_PDU is received over WTRP, the Relay Responsible List of nodes provided by the MAC layer is considered. For each of these addresses,

determine the subnet to be used for relay. Use multicast submission procedure for this list of addresses, with the broadcast sender used as original sender.

4. When a Broadcast R_PDU is received over a subnet that is not WTRP, as list of nodes is generated from the broadcast/multicast address, with the local node and original sender removed, For each of these addresses, determine the subnet to be used for relay. Use multicast submission procedure for this list of addresses, with the broadcast sender used as original sender.

5. When a Multicast R_PDU is received, the following procedure is followed.

    a. If the original sender is the local node, the PDU is discarded and processing stops.
    b. The S_PDU is delivered locally following standard procedures.
    c. Consider the list of destination addresses included in the Multicast R_PDU. If the subnet is WTRP, consider the Relay Responsible List, and eliminate all addresses that are not in this list. If there are no addresses remaining, processing stops.
    d. Pass the PDU for relay using the multicast submission procedure with this list of addresses and the original sender.

In the event of any local failure of a directly submitted R_PDU, error information from the CAS layer ispassed up to the SIS layer.

In the event of any failure of a relayed R_PDU the error must be handled by the routing sublayer. TTL will be associated with any received R_PDU. In the event of TTL expiry, the R_PDU is discarded. In the event of any other error, the R_PDU is resubmitted. The routing calculation must be repeated for the destination, as the preferred next hop may have changed since the original submission.

R.6.5. **Updating Routing**

An node connected to multiple subnets **shall** use the Routing Update R_PDU to communicate routing information to each directly connected peer node. Routing information is provided on the nodes that the local node can reach, but which are not connected to the subnet on which the Routing Update is being sent. Routing information that needs to be shared will only arise either when a node is connected to multiple Subnets, and relates either to nodes on the other subnet or routing derived from received routing updates. This information is shared at intervals or whenever routing information changes. Note that routing update is not needed on a single WTRP network.

On reception of a Routing Update R_PDU, the node must update its local configuration of reachable nodes.
The default approach to sharing Routing Update R_PDUs on a subnet is to use ARQ communication to each node on the subnet. This will ensure that full routing information is shared about all reachable nodes to all nodes.

For an Annex K (CSMA) network or a WTRP network where all nodes are directly reachable, Routing Update R_PDUs may be shared using non-ARQ broadcast.

ANNEX S SIS ACCESS PROTOCOL (Mandatory)

This annex defines a protocol for clients to access the STANAG 5066 Subnet Interface Service. This enables multiple independent clients to easily share and multiplex over a single channel.

S.1. Accessing the Subnet Interface Service

S.1.1. Changes in This Edition

The functional differences between this specification and Edition 3 are set out in Section S.7.

This is a new annex, taking Edition 3 elements from Annex A and Annex F. It brings this core STANAG 5066 protocol into a single specification to improve modularity and clarity. Key components:

1. Service Encoding. The encoding of service primitives is only used by the SIS Access Protocol, so is specified in this Annex. In Edition 3, it was specified as Section A.3. It is therefore now specified in Annex S as part of the SIS Access Protocol specification.
2. Local Services (S_SUBNET_AVAILABILITY; S_DATA_FLOW_ON/OFF; S_MANAGEMENT_MSG; S_KEEPALIVE) are specified in this Annex. In Edition 3, they were specified in Section A.2.
3. The protocol structure of this document was specified as F.16 in Edition 3.

The following services specified in Edition 3 are optional and deprecated in Edition 4 Annex A (SIS). Their use in SIS Access Protocol is specified as follows:

1. Hard Links. These are now deprecated and encoding is specified in Section S.6.1.
2. Expedited data. This service is now deprecated and encoding is specified in Section S.6.2.

S.1.2. **Overview**

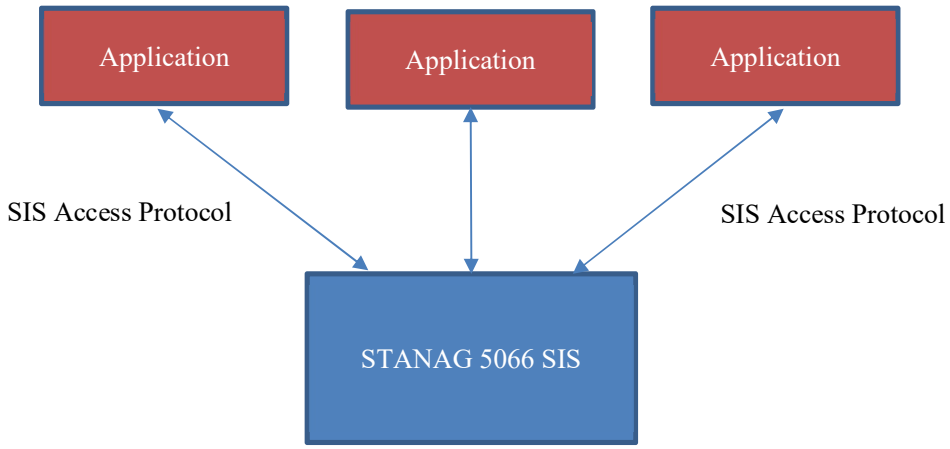


Figure S-1: SIS Access Protocol Model

A client-server relationship can be used to govern the interaction between the HF subnetwork and the users of the subnetwork, as shown in Figure S-1. The users (clients) request the services provided by the HF subnetwork (server). The service provided by the server is application independent and common to all clients irrespective of the task they may perform.

Clients are attached to the Subnetwork Interface Sublayer at Subnetwork Access Points (SAPs). There can be multiple clients simultaneously attached to the Subnetwork Interface Sublayer. Each SAP is identified by its SAP Identifier (SAP ID)1. The SAP ID is a number in the range 0-15; hence there can be a maximum of 16 clients attached to the Subnetwork Interface Sublayer of a single node.

The SIS Access protocol specified in this annex allows a client to access the Subnet Interface Service. There can be one client for each SAP, which enables multiplexing of multiple independent clients.

The SIS service specified in Annex A allows clients to access the SIS service by any mechanism. The mechanism specified in this annex is mandatory, to enable independent interoperable integration between STANAG 5066 severs and applications operating over STANAG 5066.

S.2. Mapping onto TCP

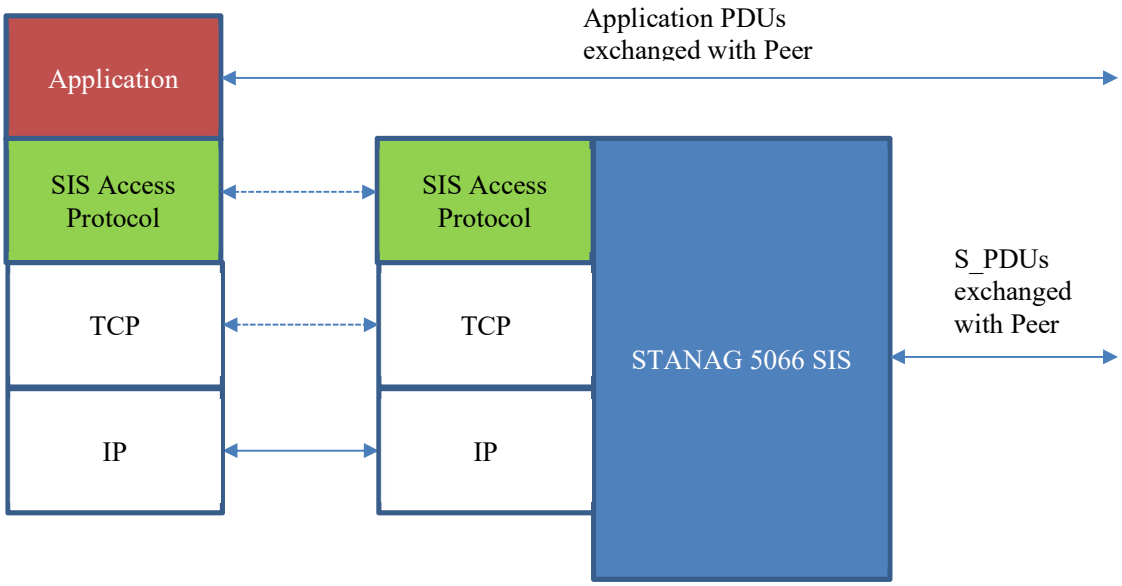


Figure S-2: SIS Access Protocol Mapping onto TCP

Figure S-2 shows how the SIS Access Protocol maps onto TCP/IP to connect an application to a SIS service. The SIS service communicates with its peer using S_PDUs. The application operating over SIS communicates with its peer using PDUs specific to the application. The SIS Access Protocol is a local protocol which does not have end to end interactions. It simply enables an application to interact with the local SIS service.

STANAG 5066 Annex A defines the core SIS services, which are extended by additional management and flow control services specified in this annex. Each of these service primitives is defined in a manner that is simple flow of data in one direction between application and SIS. This annex defines a framed encoding of each primitive, such that it can be mapped directly onto a TCP stream. This enables the application to access the SIS service over a TCP stream.

The Default SIS Port Number for the Raw SIS Socket Server **shall** be the decimal number '5066', a number registered with the Internet Assigned Number Authority (IANA) for this purpose. STANAG 5066 implementations, both clients and subnetworks, **should** be configurable to use any other valid port number as the default.

A SIS Access Protocol Server shall listen on the Default SIS Port Number for connection requests from clients. The Server shall accept sixteen connections simultaneously, so that every SAP can be served.

S_PRIMITIVES **shall** be sent over the SIS Access Protocol without any separating

characters or framing data other than the generic S_PRIMITIVE encoding elements defined in this annex. No other messages between client and subnetwork **shall** be sent. Implementation-dependent communication between a client and subnetwork over the Raw SIS-Socket-Server Interface for the purposes of system management **may** be encapsulated within the S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT _MSG_INDICATION primitives defined in this annex. [NB: This does not preclude implementation-dependent communication over another socket on a different port number, but such use is outside of the scope of this STANAG and is not recommended.]

S.3. Core Services

The primary goal of the SIS protocol is to support client access to the core SIS services, which are specified as the SIS service in Annex A of STANAG 5066.

S.4. Management & Flow Control Services

In addition to the core services, there are some additional services provided by the SIS protocol that pertain to connection management of the SIS protocol link.

Table A-1. Management and Flow Control Primitives

CLIENT -> SUBNETWORK INTERFACE	SUBNETWORK INTERFACE -> CLIENT
	S_SUBNET_AVAILABILITY (Subnet Status, Reason)
	S_DATA_FLOW_ON( )
	S_DATA_FLOW_OFF ( )
S_MANAGEMNT _MSG_REQUEST (MSG TYPE, MSG BODY)	S_MANAGEMENT_MSG_INDICATION (MSG TYPE, MSG BODY)
S_KEEP_ALIVE ( )	S_KEEP_ALIVE ( )

Table A-1 summarizes primitives to specify management and flow control operations, which extend the core SIS service primitives specified in Annex A. The details of each of these primitives is set out below.

S.4.1. Interface Flow Control Primitives: S_DATA_FLOW_ON and S_DATA_FLOW_OFF

Name :

S_DATA_FLOW_ON
S_DATA_FLOW_OFF

Arguments :
NONE
Direction :
Subnetwork Interface-> Client

Description :

The S_DATA_FLOW_ON and S_DATA_FLOW_OFF primitives **shall** (1) be issued by the Subnetwork Interface Sublayer to control the transfer of U_PDUs submitted by a client.

On receipt of an _DATA_FLOW_OFF primitive, the client **shall** (2) cease transferring U_PDUs over the interface.

Transfer over the interface of U_PDUs by the client **shall** (3) be enabled following receipt of an S_DATA_FLOW_ON primitive.

The Subnetwork Interface Sublayer can use these two primitives (or other mechanisms) to control the flow of data from locally attached clients. U_PDUs from an attached client to which the S_DATA_FLOW_OFF primitive has been sent may be discarded by the Subnetwork Interface Sublayer without acknowledgement, indication, or warning.

A client **shall** (4) not control the flow of data *from* the subnetwork by any mechanism, explicit or implicit.

All clients **shall** (5) be ready to accept at all times data received by the HF Node to which it is bound; clients not following this rule may be disconnected by the node.

S.4.2. S_MANAGEMENT_MSG_REQUEST Primitive

Name :

S_MANAGEMENT_MSG_REQUEST

Arguments :

1. MSG TYPE
2. MSG BODY

Direction :

Client-> Subnet Interface

Description :

The S_MANAGEMENT_MSG_REQUEST primitive **shall** (1) be issued by a client to submit a “Management” message to the Subnetwork.

The complex argument MSG may be implementation dependent and is not specified in this version of STANAG 5066. At present, a minimally compliant HF subnetwork implementation **shall** (2) be capable of receiving this primitive, without further requirement to process its contents.

Depending on the value of the complex argument *MSG*, this primitive can take the form of a Command (e.g. Go-To-EMCON, Go-Off-Air, etc.) or of a Request (e.g. Request-For-Subnetwork-Statistics, Request-For-Connected-client-Information, etc.).

Note that this primitive is not intended to allow for the transmission of management coordination messages over the air. This is an interaction between peer subnet management clients and as such shall be accomplished using the UNIDATA primitive defined elsewhere in this annex.

S.4.3. S_MANAGEMENT_MSG_INDICATION Primitive

Name :

S_MANAGEMENT_MSG_INDICATION

Arguments :

1. MSG TYPE
2. MSG BODY

Direction :

Subnetwork Interface-> Client

Description :

The S_MANAGEMENT_MSG_INDICATION primitive **shall** (1) be issued by the Subnetwork to send a "Management" message to a client.

The complex argument MSG may be implementation dependent and is not specified in this version of STANAG 5066. At present, a minimally compliant client **shall** (2) be capable of receiving this primitive, without further requirement to process its contents.

As implementation options, the complex argument *MSG* could take several values such as: Subnetwork- Statistics, Connected-client-Information, etc. This primitive could be issued either in response to a S_MANAGEMENT_MSG_REQUEST or asynchronously by the Subnetwork.

S.4.4. S_KEEP_ALIVE Primitive

Name :

S_KEEP_ALIVE

Arguments :

NONE

Direction :

Client-> Subnetwork
Interface Subnetwork
Interface-> Client

Description :

The S_KEEP_ALIVE primitive can be issued as required (e.g. during periods of inactivity) by the clients and/or the Subnetwork Interface to sense whether the physical connection between the client and the Subnetwork is alive or broken. This primitive may be redundant if the implementation of the physical connection provides an implicit mechanism for sensing the status of the connection.

A minimally compliant implementation of a client or subnetwork interface is not required to generate the S_KEEP_ALIVE primitive except in response to the receipt of an S_KEEP_ALIVE primitive.

When the S_KEEP_ALIVE Primitive is received, the recipient (i.e, client or Subnetwork Interface) **shall**
respond with the same primitive within 10 seconds.

If a reply is not sent within 10 seconds, no reply **shall** (2) be sent.

A client or Subnetwork Interface **shall** (3) not send the S_KEEP_ALIVE Primitive more frequently than once every 120 seconds to the same destination.

S.4.5. S_SUBNET_AVAILABILITY Primitive

Name :

S_SUBNET_AVAILABILITY

Arguments :

1. Node Status
2. Reason

Direction :

Subnetwork Interface-> Client

Description:

The S_SUBNET_AVAILABILITY primitive may be sent asynchronously to all or selected clients connected to the Subnetwork Interface Sublayer to inform them of changes in the status of the node to which they are attached. For example, clients can be informed using this primitive that available resources (e.g., bandwidth) have been temporarily reserved for other clients. Alternatively, this primitive could be used to inform clients that the node has entered an EMCON state and as a result they should only expect to receive Data and will not be allowed to transmit data.

The contents of this primitive are implementation dependent and not specified in this version of STANAG 5066. At present, a minimally compliant client implementation **shall** (1) be capable of receiving this primitive, without further requirement to process its contents.

As implementation options, the *Node Status* argument could specify the new Status of the node. Possible values of this argument could be ON, OFF, Receive-Only, Transmit-Only-to-Specific-Destination- Node/SAP, etc.

Node Status	Value
OFF	0
ON	1
Receive-Only	2
Half-Duplex	3
Full-Duplex	4
Transmit-Only	5

If the Subnetwork Status is other than ON, the *Reason* argument explains why. Values of this argument shall be as specified below.

The value assigned to each Node Status **shall** be used to represent the reason in SIS Access Protocol (Annex S).

Reason	Value
unspecified	0
Local Node in EMCON	1
Reserved	2

The value assigned to each reason **shall** be used to represent the reason in SIS Access Protocol (Annex S).

S.5. Encoding of Primitives

The encoding of the S_Primitives for communication using the protocol specified in this annex **shall** (1) be in accordance with text and figures in the subsections below.

S.5.1. Generic Field Encoding Requirements

Unless noted otherwise, the bit representation for argument values in an S_Primitive **shall** (1) be encoded into their corresponding fields in accordance with CCITT V.42, 8.1.2.3, which states that:

- when a field is contained within a single octet (i.e, eight bit group), the lowest bit number of the field **shall**(2) represent the lowest-order (i.e., least-significant-bit) value;
- when a field spans more than one octet, the order of bit values within each octet **shall** (3) progressively decrease as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.

The 4-byte address field in the S_primitives **shall** (4) carry the 3.5-byte address and address-size information defined in Section Node ADDRESS Encoding for all PrimitivesS.5.16.1. The lowest order bit of the address shall be placed in the lowest order bit position of the field (generally bit 0 of the highest byte number of the field), consistent with the mapping specified in Section C.3.2.6 for D_PDUs.

S.5.2. S_Primitive Generic Elements and Format

As shown in Figure A-1(a), all primitives **shall** (1) be encoded as the following sequence of elements:

- a two-byte S_Primitive preamble field, whose value is specified by the 16-bit Maury-Styles sequence below;

- a one-byte version-number field;

- a two-byte Size_of_Primitive field;

- a multi-byte field that contains the encoded S_Primitive.

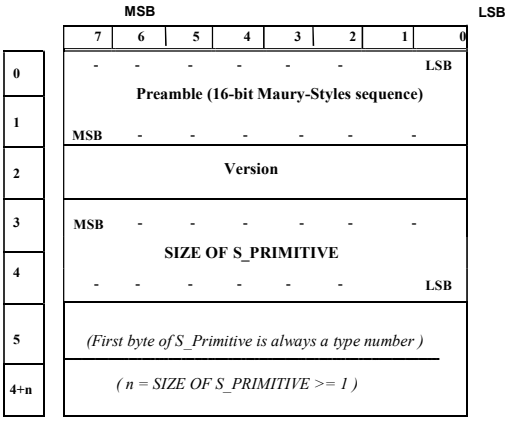


Figure S-3: Element-Sequence *Encoding of "S_" Primitives*

The S_Primitive preamble field **shall** (2) be encoded as the 16-bit Maury-Styles sequence shown below, with the least significant bit (LSB) transmitted first over the interface:

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

i.e., with the multi-byte S_Primitive field represented in hexadecimal form as 0xEB90, the least-significant bits of the sequence **shall** (3) be encoded in the first byte (i.e, byte number 0) of the preamble field and the most significant bits of the sequence **shall** (4) be encoded in the second byte (i.e, byte number 1) of the preamble field as follows:

	MSB 7	6	5	4	3	2	1	LSB 0	
0	1	0	0	1	0	0	0	0	0x90
1	1	1	1	0	1	0	1	1	0xEB

Figure S-4: Encoding of Maury-Styles Preamble-Sequence in "S_" Primitives

[***Note***: This encoding of the Maury-Styles preamble sequence is an exception to the general requirement of section S.5.1 for field encoding.]

Following the Maury-Styles sequence, the next 8 bit (1-byte) field **shall** (5) encode the 5066 version number. For this version of STANAG 5066, the version number **shall** (6) be all zeros, i.e, the hexadecimal value 0x00, as shown in Figure S-3.

The next 16 bit (two-byte) field **shall** (7) encode the size in bytes of the S_primitive-dependent field to follow, exclusive of the Maury-Styles sequence, version field, and this size field. The LSB of the of the size value **shall** be mapped into the low order bit of the low-order byte of the field as shown in Figure S-3.

Unless specified otherwise, the order of bit transmission for each byte in the encoded S_Primitive **shall** (9) be as described in CCITT V.42 paragraph 8.1.2.2, which specifies the least significant bit (LSB, bit 0 in the figures below) of byte 0 **shall** (10) be transmitted first.

The sixth byte (i.e., byte number 5) of the sequence **shall** (11) be the first byte of the encoded primitive and **shall** be equal to the S_Primitive type number, with values encoded in accordance with the respective section that follows for each S_primitive

The remaining bytes, if any, in the S_Primitive **shall** (13) be transmitted sequentially, also beginning with the LSB of each byte, in accordance with the respective section that follows for each S_primitive.

In the subsections that follow, any bits in a S_Primitive that are specified as NOT USED **shall** (13) be encoded with the value “0” unless specified otherwise for the specific S_Primitive being defined.

S.5.3. **S_BIND_REQUEST Encoding**

The S_BIND_REQUEST primitive **shall** (1) be encoded as a four-byte field as follows:

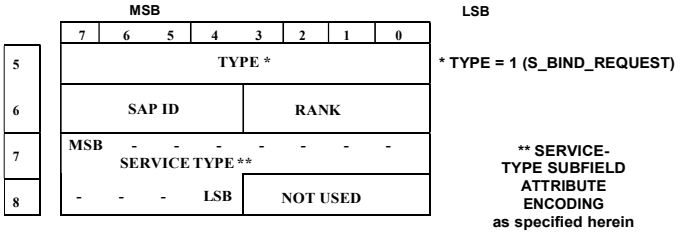
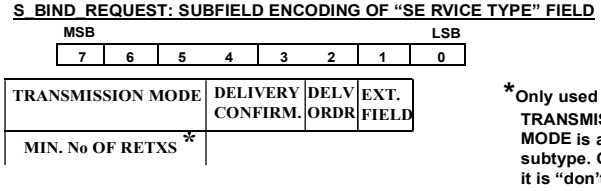


Figure S-5: Encoding of S_BIND_REQUEST Primitive

The S_BIND_REQUEST SERVICE-TYPE field **shall** (2) be encoded as five subfields as follows:



TRANSMISSION MODE	DELIVERY CONFIRMATION	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore Service type field. 0001 = ARQ 0010 = Non-ARQ (e.g., Broadcast) 0011 = Non-ARQ WITH ERRORS	00 = No Confirmation 01 = Node Delivery Conf. 10 = Client Delivery Conf. 11 = Not defined	0 = In-order Delivery 1 = As they arrive	0 = No extended field 1 = Extended field follows

Figure S-6: Sub-field Attribute Encoding of S_BIND_REQUEST SERVICE-TYPE field.

Argument : SERVICE
TYPE Primitive :
S_BIND_REQUEST

The SERVICE TYPE argument **shall** (3) specify the default type of service requested by the client. This type of service **shall** (4) apply to any U_PDU submitted by the client until the client unbinds itself from the node, unless overridden by the DELIVERY MODE argument of the U_PDU. A client **shall** (5) change the default service type only by unbinding and binding again with a new S_BIND_REQUEST.

The RANK value was present in Edition 3 of STANAG 5066, but is not used in this edition. It should be set to zero on transmission and ignored on reception.

The SERVICE TYPE argument is complex, consisting of a number of attributes encoded as sub-fields. Although the exact number of attributes and their encoding is left for future definition and enhancement using the Extended Field attribute, the following attributes are mandatory:

1. *Transmission Mode for the Service.* --- ARQ or Non-ARQ Transmission Mode **shall**(6) be specified, with one of the Non-ARQ submodes if Non-ARQ was requested. A value of "0" for this attribute **shall**(7) be invalid for the SERVICE TYPE argument when binding. Non-ARQ transmission can have submodes such as: *Error-Free-Only* delivery to destination client, delivery to destination client even with *some* errors.

2. *Data Delivery Confirmation for the Service* --- The client **shall** (8) request one of the Data Delivery Confirmation modes for the service. There are three types of data delivery confirmation:
   - None
   - Node-to-Node Delivery Confirmation
   - Client-to-Client Delivery Confirmation

   The client can request explicit confirmation, i.e, Node-to-Node or Client-to-Client, from the Subnetwork to provide indication that its U_PDUs have been properly

delivered to their destination. Explicit delivery confirmation **shall** (9) be requested only in combination with ARQ delivery.

[Note: The Node-to-Node Delivery Confirmation does not require any explicit peer-to-peer communication between the Subnetwork Interface Sublayers and hence it does not introduce extra overhead. It simply uses the ACK (ARQ) confirmation provided by the Data Transfer Sublayer. Client-to-Client Delivery Confirmation requires explicit peer-to-peer communication between the Sublayers and therefore introduces overhead. It should be used only when it is absolutely critical for the client to know whether or not its data was delivered to the destination client (which may, for instance, be disconnected).]

3. *Order of delivery of any U_PDU to the receiving client.* --- A client **shall** (10) request that its U_PDUs are delivered to the destination client "in-order" (as they are submitted) or in the order they are received by the destination node.

4. *Extended Field* --- Denotes if additional fields in the SERVICE TYPE argument are following; at present this capability of the SERVICE TYPE is undefined, and the value of the Extended Field Attribute **shall** (11) be set to "0".

5. *Minimum Number of Retransmissions* --- This argument **shall** (12) be valid if and only if the Transmission Mode is a Non-ARQ type. If the Transmission Mode is a Non-ARQ type, then the subnetwork **shall** (13) retransmit each U_PDU the number of times specified by this argument. This argument may be "0", in which case the U_PDU is sent only once.

   [Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]

S.5.4. S_UNBIND_REQUEST Encoding

The S_UNBIND_REQUEST primitive **shall** (1) be encoded as a one-byte field as follows:

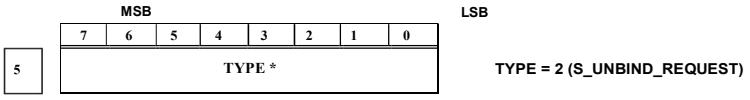


Figure S-7: Encoding of S_UNBIND_REQUEST Primitive

S.5.5. **S_BIND_ACCEPTED Encoding**

The S_BIND_ACCEPTED primitive **shall** (1) be encoded as a four-byte field as follows:

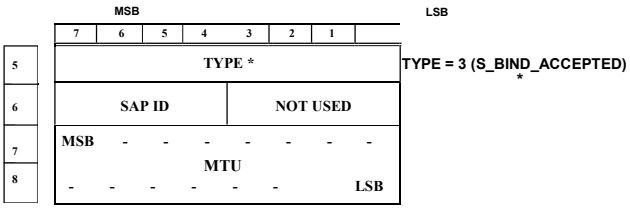


Figure S-8: Encoding of S_BIND_ACCEPTED Primitive

S.5.6. **S_BIND_REJECTED Encoding**

The S_BIND_REJECTED primitive **shall** (1) be encoded as a two-byte field as follows:

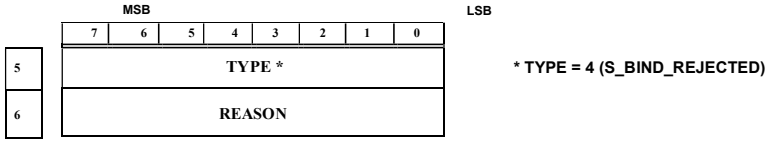


Figure S-9: Encoding of S_BIND_REJECTED Primitive

The Reason **shall** be set to a value specified in A.2.2.4 of Annex A.

S.5.7. **S_UNBIND_INDICATION Encoding**

The S_UNBIND_INDICATION primitive **shall** (1) be encoded as a two-byte field as follows:

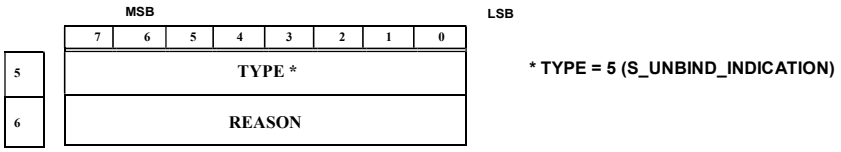


Figure S-10: Encoding of S_UNBIND_INDICATION Primitives

The Reason **shall** be set to a value specified in A.2.2.5 of Annex A.

S.5.8. **S_SUBNET_AVAILABILITY Encoding**

The S_SUBNET_AVAILABILITY primitive **shall** (1) be encoded as a three-byte field as follows:

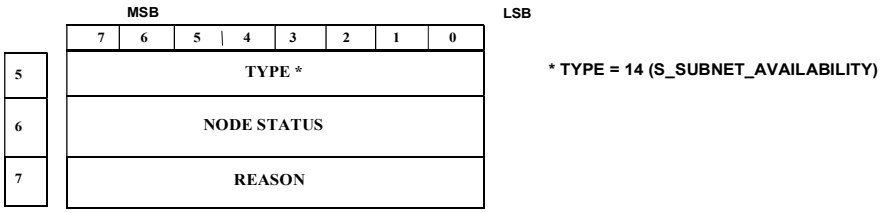


Figure S-11: Encoding of S_SUBNET_AVAILABILITY Primitives.

The encoding of the NODE STATUS and REASON fields is implementation dependent.

S.5.9. **S_DATA_FLOW_ON and S_DATA_FLOW_OFF Encoding**

The S_DATA_FLOW_ON and S_DATA_FLOW_OFF primitives **shall** (1) be encoded as one-byte fields as follows:

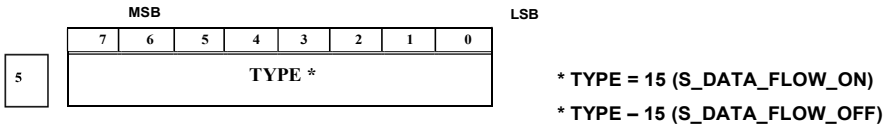


Figure S-12: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

S.5.10. **S_KEEP_ALIVE Encoding**

The S_KEEP_ALIVE primitive **shall** (1) be encoded as a one-byte field as follows:

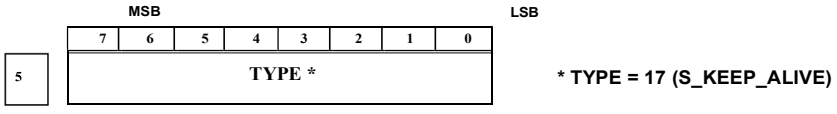


Figure S-13: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

S.5.11. S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_ MSG_INDICATION Encoding

The S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_ INDICATION primitives **shall** (1) be encoded as implementation-dependent variable-length fields as follows:

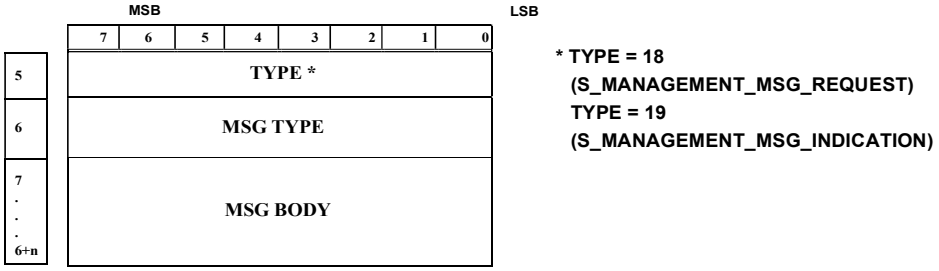


Figure S-14: Encoding of S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_INDICATION Primitives.

The encoding of the MSG TYPE and MSG BODY fields is implementation dependent.

S.5.12. S_UNIDATA_REQUEST Encoding

The S_UNIDATA_REQUEST primitive **shall** (1) be encoded as a variable-length field as follows:

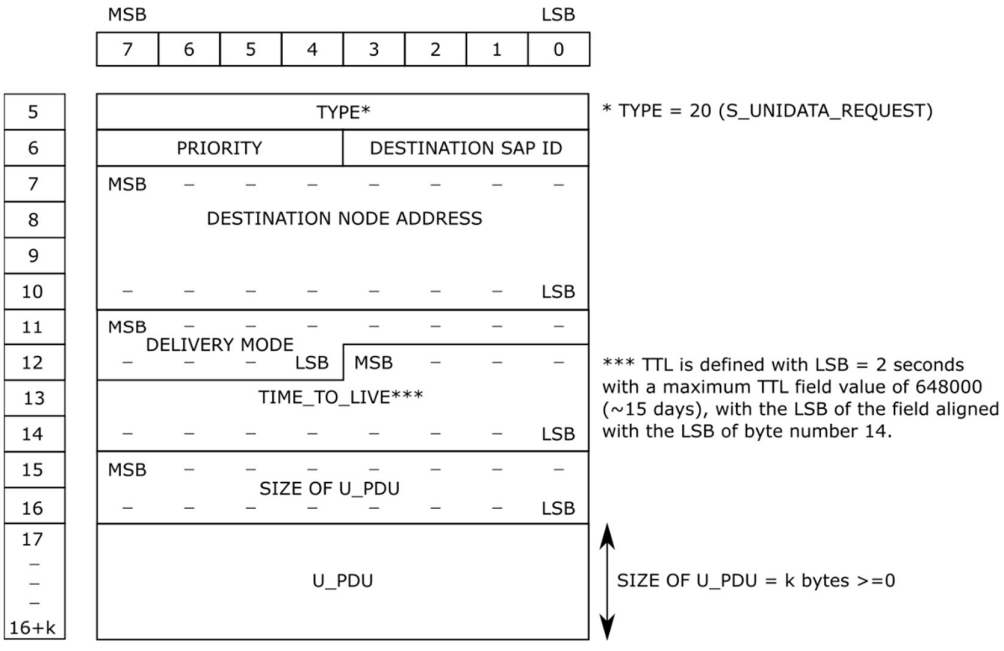


Figure S-15: Encoding of S_UNIDATA_REQUEST Primitives.

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in SectionS.5.16.1.

The DELIVERY MODE field **shall** (3) be encoded as specified in Section S.5.16.2.

S.5.13. **S_UNIDATA_INDICATION Encoding**

The S_UNIDATA_INDICATION primitive **shall** (1) be encoded as a variable-length field as follows:

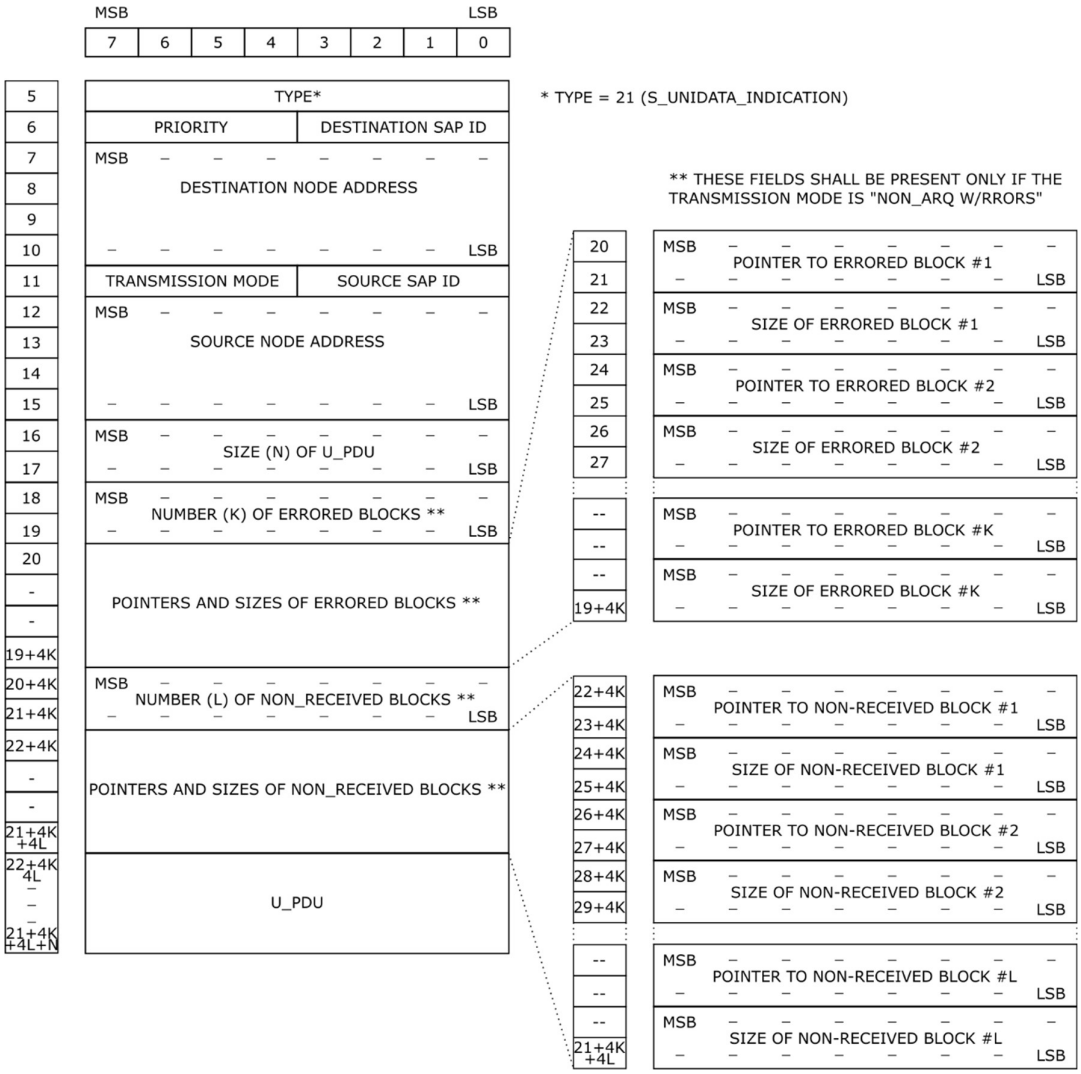


Figure S-16: Encoding of S_UNIDATA_INDICATION Primitives

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in Section S.5.16.1.

The TRANSMISSION MODE field **shall** (3) be encoded as specified in Section S.5.16.3.

S.5.14. **S_UNIDATA_REQUEST_CONFIRM Encoding**

The S_UNIDATA_REQUEST_CONFIRM primitive **shall** (1) be encoded as a variable-length field as follows:

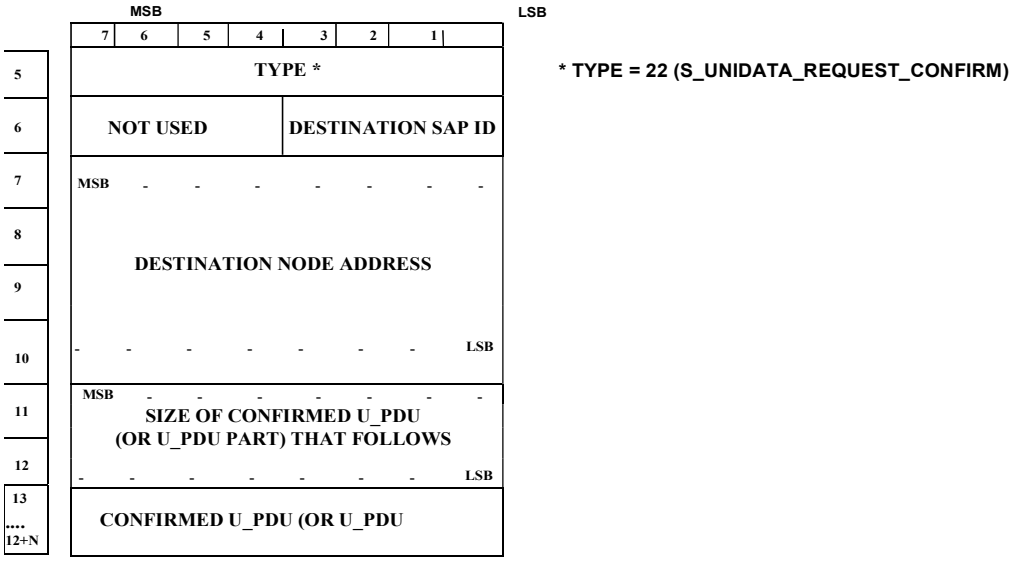


Figure S-17: Encoding of S_UNIDATA_REQUEST_CONFIRM Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section S.5.16.1.

S.5.15. **S_UNIDATA_REQUEST_REJECTED Encoding**

The S_UNIDATA_REQUEST_REJECTED primitive **shall** (1) be encoded as a variable-length field as follows:

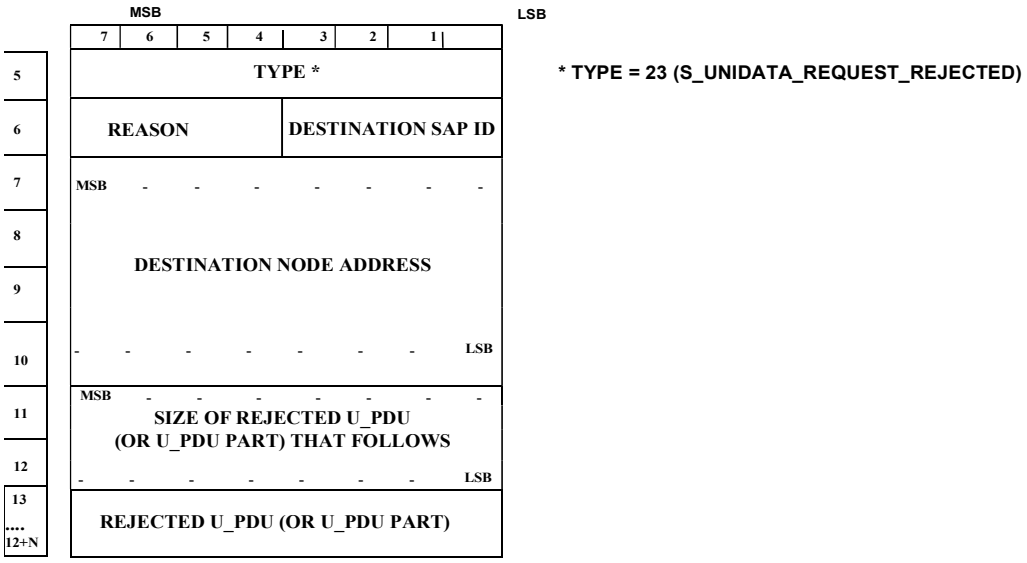


Figure S-18: Encoding of S_UNIDATA_REQUEST_REJECTED Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section S.5.16.1.

The Reason **shall** be set to a value specified in A.2.2.8 of Annex A.

S.5.16. **Additional S_Primitive Encoding Requirements: Encoding of Common Fields**

In order to clarify some of the procedures and tasks executed by the sublayers, additional details concerning some of the arguments of the Primitives described in previous sections are provided below.

S.5.16.1. Node ADDRESS Encoding for all Primitives

Arguments : SOURCE NODE ADDRESS, DESTINATION NODE ADDRESS, or REMOTE NODE ADDRESS
Primitives : ALL “UNIDATA” primitives.

For reduced overhead in transmission, node addresses **shall** (1) be encoded in one of several

formats that are multiples of 4-bits (“half-bytes”) in length, as specified in Figure S-19.

Addresses that are encoded as Group node addresses **shall** (2) only be specified as the Destination Node address of Non-ARQ PDUs.

Destination SAP IDs and destination node addresses of ARQ PDUs and source SAP IDs and source node addresses of all PDUs **shall** (3) be individual SAP IDs and individual node addresses respectively.

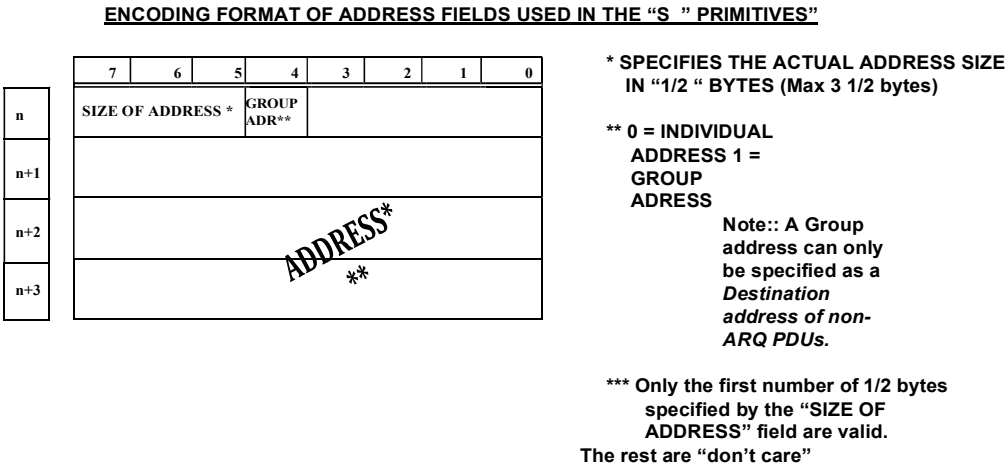


Figure S-19: Encoding of Address Fields in S_Primitives.

S.5.16.2. Delivery-Mode Encoding for the S_UNIDATA_REQUEST Primitive

Argument : DELIVERY MODE
Primitive : S_UNIDATA_REQUEST

The DELIVERY MODE is a complex argument consisting of a number of attributes, as specified here. The DELIVERY MODE argument **shall** (1) be encoded as shown in Figure S-20.

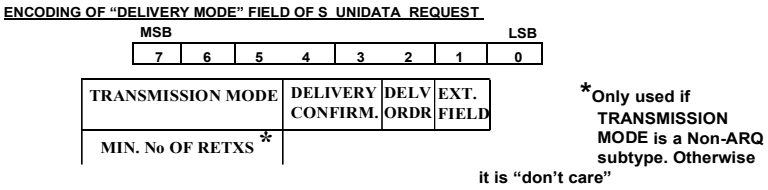
The value of the DELIVERY MODE argument can be “DEFAULT”, as encoded by the Transmission Mode attribute. With a value of “DEFAULT”, the delivery mode for this U_PDU **shall** (2) be the delivery mode specified in the *Service Type* argument of the S_BIND_REQUEST. A non-DEFAULT value **shall** (3) override the default settings of the Service Type for this U_ PDU.

The attributes of this argument are similar to those described in the *Service Type* argument of the S_BIND_REQUEST:

1. *Transmission Mode of this U_PDU.* --- ARQ or Non-ARQ Transmission can be requested. A value of “0” for this attribute **shall** (4) equal the value “DEFAULT” for the Delivery Mode. If the DELIVERY MODE is “DEFAULT”, all other attributes encoded in the argument **shall** (5) be ignored.
2. *Data Delivery Confirmation for this PDU* --- None, node-to-node, or client-to-client.

3. *Order of delivery of this PDU to the receiving client.* --- A client may request that its U_PDUs are delivered to the destination client "in-order" (as they are submitted) or in the order they are received by the destination node.
4. *Extended Field* --- Denotes if additional fields in the DELIVERY MODE argument are following; at present this capability of the DELIVERY MODE is undefined, and the value of the Extended Field Attribute **shall** (6) be set to "0".
5. *Minimum Number of Retransmissions* --- This argument **shall** (7) be valid if and only if the Transmission Mode is a Non-ARQ type or sub-type. If the Transmission Mode is a Non-ARQ type or subtype, then the subnetwork **shall** (8) retransmit each U_PDU the number of times specified by this argument. This argument may be "0", in which case the U_PDU is sent only once.

   [Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]



TRANSMISSION MODE	DELIVERY CONFIRMATION	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore Service type field. 0001 = ARQ 0010 = Non-ARQ (e.g., Broadcast) 0011 = Non-ARQ WITH ERRORS	00 = No Confirmation 01 = Node Delivery Conf. 10 = Client Delivery Conf. 11 = Not defined	0 = In-order Delivery 1 = As they arrive	0 = No extended field 1 = Extended field follows

Figure S-20: Encoding of the Delivery Mode field in the S_UNIDATA_REQUEST and primitives

S.5.16.3. TRANSMISSION-MODE Encoding for the S_UNIDATA_INDICATION Primitive

Argument: TRANSMISSION-MODE
S_Primitives: S_UNIDATA_INDICATION

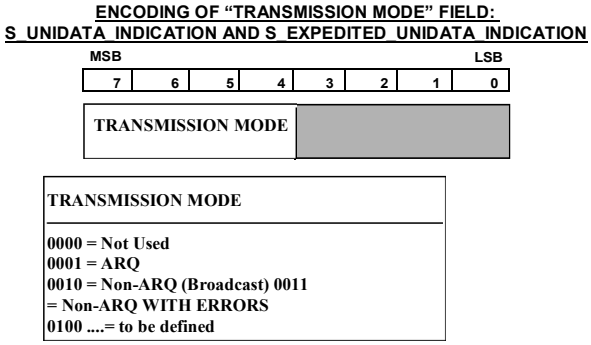


Figure S-21: Encoding of Transmission Mode Field in S_UNIDATA_INDICATION primitive.

The subnetwork notifies a client of the transmission-mode used to deliver a U_PDU Argument with the TRANSMISSION-MODE argument. The TRANSMISSION-MODE argument in the S_UNIDATA_INDICATION Primitives **shall** (1) be encoded as shown in Figure S-21.
[Note: The unused bits in this argument are allocated to the SOURCE SAP_ID argument encoding for the S_UNIDATA_INDICATION Primitive.]

S.6. 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.

S.6.1. Hard Links

This optional deprecated service is defined as the HARD LINKS profile option. The service is specified in Annex A.

The encoding of Hard Links protocol Elements in the SIS Access Protocol is specified in Edition 3 Annex A, Section A.3. Use of Rank may be necessary as part of the Hard Links service.

S.6.2. Expedited Data

This optional deprecated service is defined as the EXPEDITED DATA profile option. The service is specified in Annex A.

The encoding of Expedited Data protocol Elements in the SIS Access Protocol is specified in Edition 3 Annex A, Section A.3.

S.7. Changes in Edition 4

This Annex is built from information in Edition 3 Annexes A and F. It supports the services defined in Annex A, and the protocol specified is unchanged from Edition 3.

It does not specify protocol for the hard link and expedited services, which are defined only in Edition 3.

ANNEX T STANAG 5066 TRANSEC CRYPTO SUBLAYER USING AES AND OTHER PROTOCOLS (Optional)

The TRANSEC Crypto Sublayer is an optional sublayer that provides TRANSEC. It defines a framework for using arbitrary stream cryptography to provide TRANSEC. It specifies how to use Advanced Encryption Standard (AES) in this framework.

The TRANSEC specified in Annex T can be used as an alternative to synchronous serial crypto devices connected to the DTS using STANAG 5066 Annex D.

This annex is new to Edition 4 of STANAG 5066.

T.1. Overview

This annex specifies protocol for supporting a Crypto Layer between STANAG 5066 and Modem, following the STANAG 5066 model. This protocol defines a generic framework for use with different encryption algorithms, with a specific mapping for AES encryption.

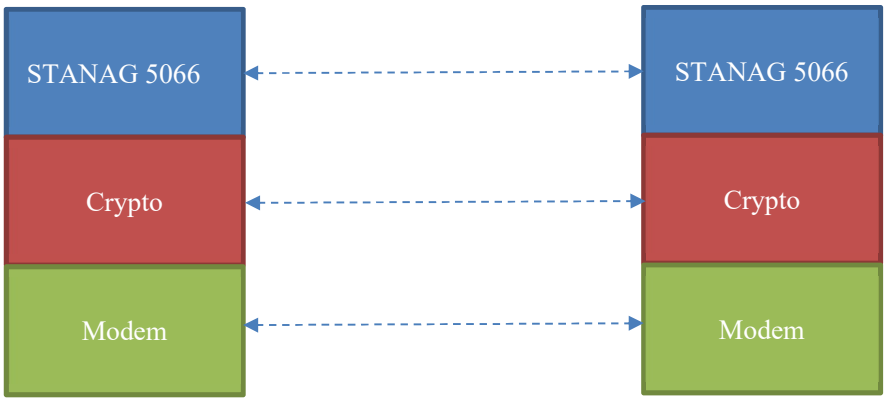


Figure T-1: STANAG 5066 TRANSEC Crypto Architecture

The STANAG 5066 architecture for use of TRANSEC Crypto is shown in Figure T-1. Crypto is used immediately above the modem, which is the lowest possible point that Crypto could be inserted. A stream crypto is used, so that modem data errors only impact that data. When crypto is synchronized, it does not introduce additional errors. This is an efficient approach.

HF traffic is easy to monitor and even the lowest layers of STANAG 5066 contain information that is of potential interest. Therefore, it makes good security sense to perform encryption at this lowest possible layer.

T.1.1. **Benefits of a Protocol Approach**

This specification introduces a protocol approach, which gives the benefits of the architecture and removes the overheads and issues associated with use of sync serial following Annex D. A number of implementation approaches are possible. An implementation approach that could be taken is:

1. Use the open TCP protocol interface specified in MIL-STD-188-110D Annex A to communicate between Crypto Layer and modem.
2. Implement the Crypto Layer framing as specified here, as part of the STANAG 5066 server. This can offer:
    a. Built in AES support.
    b. Plugin options to drive other Crypto.

T.2. Crypto Considerations

This section considers a number of crypto issues.

T.2.1. **AES**

AES (Advanced Encryption Standard) is a widely adopted US Government standard for encryption. It is widely used for commercial, government and military operation. Therefore, use of AES is specified in this annex.

T.2.2. **COMSEC and TRANSEC**

TRANSEC (Transmission Security) is the protection applied at the lowest communication level, of which Crypto between modem and STANAG 5066 is a good example. Technically, this annex specifies TRANSEC.

COMSEC (Communication Security) is the protection applied to user data. TRANSEC **may** be used to provide COMSEC. The model for use of Annex D is that COMSEC is provided by TRANSEC. This annex defines TRANSEC. It is a deployment decision as to whether this is also used for COMSEC.

T.3. Modem Service Specification

In order to understand how the Crypto Layer works, it is important consider the modem service interface.

T.3.1. **Modem Transmission**

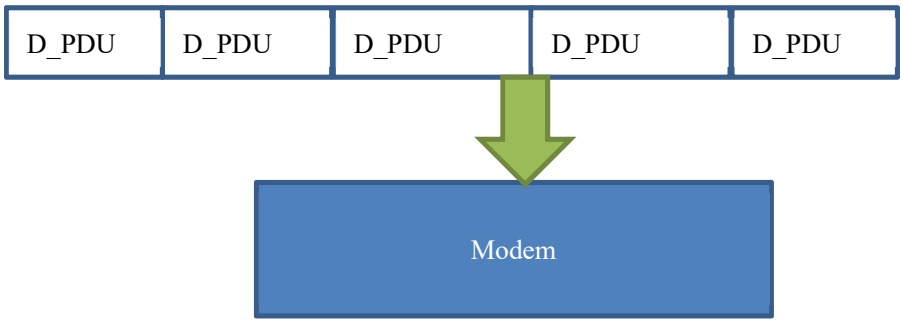


Figure T-2: Transmission of D_PDUs to a Modem

STANAG 5066 will transmit over the modem a “bounded stream” of D_PDUs as shown in Figure T-2. This is sent as a sequence of bytes, with request to start transmission implicit in the first byte. When the last byte of the last D_PDU is sent, it is marked as “end of stream”. The modem may pad after the last D_PDU in order to fill an exact number of blocks.

STANAG 5066 will also set parameters for the transmission, which will be fixed for the period of transmission and will deal with any errors reported. Parameters include: speed; interleaver; waveform; waveform-specific parameters; bandwidth.

For duplex and broadcast, transmissions may be of arbitrary length. In other cases, transmissions are limited to 127.5 seconds and each D_PDU is marked with remaining transmission time (EOT) in units of 0.5 seconds.

This information is transparent to the modem, but important for overall operation. STANAG 5066 will know the speed of modem in order to calculate data to send and to keep up with the modem. Transmission length is determined as start of transmission. However, it is desirable to defer choice of which data to send, to enable insertion of high priority data arriving after a transmission starts.

T.3.2. **Modem Reception**

Under good conditions, modem reception of data will be entirely symmetrical to transmission, and a bounded stream of D_PDUs will be provide to the receiving server. On reception with modern waveforms, most transmission parameters are determined from the transmission. Dealing with modem reception issues is key to Crypto Layer design.

T.3.3. **Determining Transmission End**

Data is often lost or corrupted during transmission. A key problem is to determine end of transmission. There are strict rules for modem transmission. A receiving modem

needs to apply heuristics to determine transmission end, particularly with poor HF conditions or aggressive choice of transmission speed.

There are two modem level mechanisms for determining end of transmission:

1. Modem Protocol. This is supported in STANAG 5069, but not older protocols. It definitively marks end of transmission.
2. EOM Marker. Two special bytes sent in the data stream. Care needs to be taken to handle the case where the "real data" includes this value. Heuristics to validate include ensuring that only "padding data" follows the EOM and that the RF signal falls off after the block is complete.

Both of these mechanisms can be "lost" due to fading or other data corruption. A modem will detect loss of RF signal. This can be an indication of transmission end or it could be a reception gap in a longer transmission. A modem will generally wait for a period before considering the transmission ended due to loss of RF. During this time, the modem will continue to send data to STANAG 5066 at "modem speed" and will maintain synchronization with the transmission (which may have ended).

The D_PDU EOT mechanism is helpful when the modem continues to receive in this way. A STANAG 5066 server can determine the end of transmission time from any single D_PDU. It will "know" that a transmission has finished, even when the modem continues to provide data (e.g., because Modem has not explicitly detected end of the RF transmission and is maintaining sync). The DTS will be able to switch to transmission, while the modem is still sending it (spurious) data.

T.3.4. **Modem Synchronizing During Transmission**

HF Waveforms start with a robust synchronization sequence, and so normal expectation is that modems synchronize at start of transmission, so that data bytes are correctly aligned over the transmission, even when there is loss and corruption.

Modem protocols can resynchronize during a transmission. This means that data can start to be received part way through a transmission. Also, a transmission can be "lost" and then a latter part of the transmission picked up as a new transmission.

Some waveforms, notably STANAG 4539 and STANAG 4285, synchronize very well during when the initial synchronization fails and it is desirable to allow for this on reception.

STANAG 5069 does not synchronize well during a transmission, if the initial synchronization fails. This means that when STANAG 5069 is used, it may be desirable to use longer (and more robust) pre-ambles and **may** be desirable to avoid overly long transmissions.

T.4. Crypto Layer

T.4.1. Service Interfaces

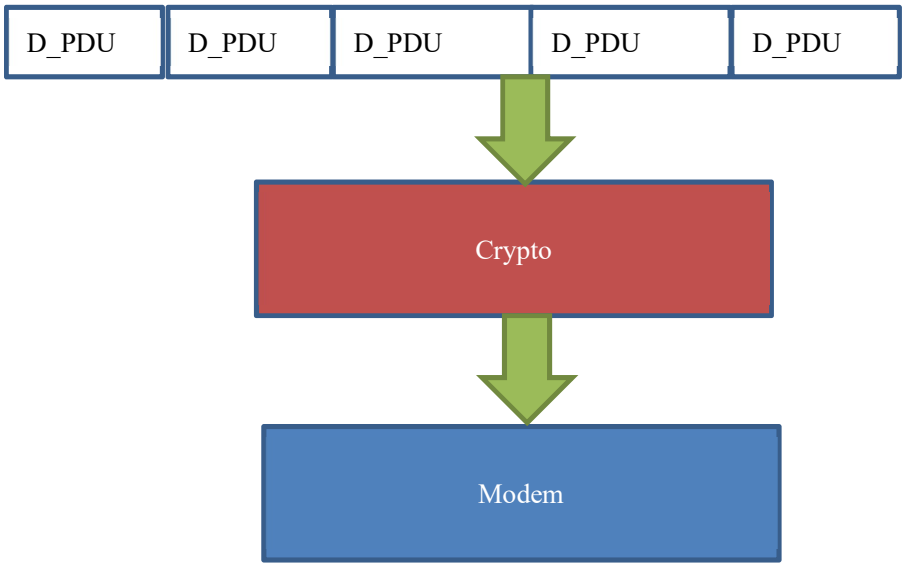


Figure T-3: STANAG 5066 Interface to Crypto

The basic service model of this protocol is that the data interface to and from the crypto layer is essentially the same and shown in Figure T-3. This is in line with STANAG 5066 Annex D.

STANAG 5066 needs to be aware of the protocol overheads of the Crypto Layer, so that it can correctly calculate transmission lengths. It **may** also choose to align D_PDU boundaries to modem block boundaries, which can be done precisely with this protocol stack.

T.4.2. Crypto Mapping & Counter Mode

The Crypto needs to operate as a stream crypto. Most modern encryption algorithms are block-oriented, which cannot be used directly. This is addressed with Counter (CTR) Mode, which is a mechanism to provide stream encryption from a block cipher. This is now widely recognized as a secure approach.

Counter mode works by initializing the cipher with an Initialization Vector (IV) and/or a Nonce. This then generates a crypto stream of bytes with as many bytes as needed. Sender and Receiver share the IV/Nonce and so can generate identical crypto streams.

The sender will XOR the data stream with the crypto stream to produce a transmission stream, which is sent between the modems. The receiver can then XOR the received

transmission stream with the crypto stream to restore the data stream sent by the peer STANAG 5066 server. This received stream may have data corruption due to modem level HF errors.

The strict synchronization of modem data transfer ensures that the streams remain aligned over the complete transmission.

T.4.3. Crypto Initialization

This section considers crypto initialization in more detail.

T.4.3.1. General Model

There is information which needs to be shared between sender and receiver which is configured prior to the transmission. This might be an external mechanism such as pre-placed keys, or other external mechanism.

Then there is information provided for each transmission, which will typically be an IV and/or Nonce, but there could be other information for encryption mechanisms other than AES.

T.4.3.2. AES Initialization

Each transmission will provide a 2 byte reference to the AES Key/Nonce pair. This reference enables:

- Different keys may be used for different pairs of 1:1 communication and multicast/broadcast groups.
- New keys can easily be used in the event of key compromise.
- Giving keys a limited lifetime, with migration to new keys.

Management of this reference and distribution of AES Keys and Nonces is not specified in this annex.

Each transmission will include an 8 byte IV that is unique for the AES Key/Nonce pair. Uniqueness can be ensured by simple incrementing or by Linear Feedback Shift Register. The last IV used should be recorded on permanent storage, so the unique IV will be ensured in the event of restart.

T.4.3.3. Resilience over HF

Transferring the initialization information needs to address potential corruption on transfer. HF errors tend to cluster, so the best approach to resilience is to repeat the information at intervals in the data stream.

The protocol also needs to address the possibility of the initial modem synchronization not happening. This is done by use of extended information that includes:

1. A Maury-Styles two byte pair, that enables synchronization.
2. A counter to enable the amount of preceding data to be calculated. This will enable to full crypto-stream to be determined, so that the received data can be XOR'd from the correct point.

T.5. Crypto Layer Protocol

The Crypto protocol defines two PDUs. The Crypto Sync PDU is specified in Figure T-4.

	MSB 7	6	5	4	3	2	1	LSB 0
0	Version=0				Algorithm			
1	Crypto Sync							
n								

Figure T-4: Crypto Sync PDU

For this version of the protocol, Version=0. The Algorithm identifies that Algorithm used. The length and semantics of the Crypto Sync bytes are determined by the Algorithm.

AES has Algorithm=0, and the encoding specified in Figure T-5 to give an 11 byte PDU.

	MSB 7	6	5	4	3	2	1	LSB 0
0	Version=0				Algorithm=0			
1	Key/Nonce Reference							
2								
3	Initialization Vector (IV)							
10								

Figure T-5: Crypto Sync PDU for AES

The encoding in Figure T-5 has the following elements:

1. The Key/Nonce Reference is two bytes and identifies the AES Key and Nonce pair to be used.
2. The Initialization Vector is an 8 byte IV.

The Extended Crypto Sync PDU is defined in Figure T-6:

	MSB 7	6	5	4	3	2	1	LSB 0
0		Maury-Styles						
1								
2	Version=0					Algorithm		
3		Sync Counter						
4								
5		Crypto Sync						
n+4								

Figure T-6: Extended Crypto Sync PDU

The encoding in in Figure T-6 has the following elements:

1. Maury-Styles is a two byte fixed header using the STANAG 5066 Maury-Styles value (90EB) using the same format as the one for D_DPUs defined in Annex C.

2. Sync Counter indicates the number of the Extended Crypto Sync PDU. The first one is numbered 1, the second 2, and so on. For AES, this PDU is 15 bytes long.

In order to protect against loss of Crypto Sync PDU, it is repeated in the transmission. Because transmission times can vary from less than a second to many minutes and speeds from 75bps to 240 kbps, there is a high variation of size of bounded stream. Repetition of Crypto Sync PDUs needs to be close enough to ensure repetition at the slowest speeds and to provide reasonable spacing at higher speeds to provide protection against corruption of full modem block.

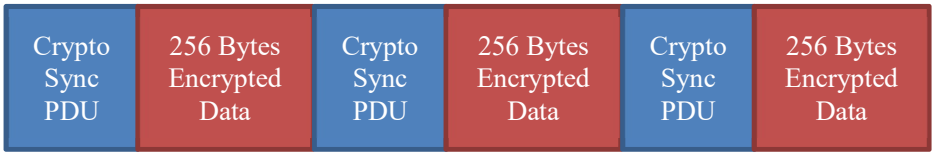


Figure T-7: Crypto Sync PDU repetition for first 4096 bytes of Data

For the first 4096 bytes of transmitted data, a Crypto Sync PDU shall inserted before every 256 bytes of encrypted data, as shown in Figure T-7. If less than 4096 bytes of encrypted data is being transmitted, the last block **may** be less than 256 bytes. Figure T-7shows the start of a block transmitted data. For AES, the Crypto Sync in this region has an overhead of 4.3%.

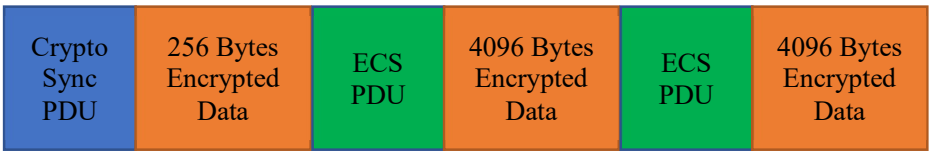


Figure T-8: Extended Crypto Sync PDU repetition

If more that 4096 bytes of encrypted data is transferred, the initial 4096 bytes are transferred as shown in Figure T-7. After this, data is sent in 4096 byte blocks, with each block preceded by an Extended Crypto Sync PDU. The last block **may** be less than 256 bytes.

Figure T-8 shows the last 256 bytes of the first 4096 bytes of data, followed by two blocks of 4096 bytes. For this extended region, AES Crypto Sync overhead is 0.4%.

ANNEX U IP CLIENT (Optional)

This Annex specifies operation of IP (the Internet Protocol) over STANAG 5066, covering both IPv4 (RFC 791) and IPv6 (RFC 8200). This enables support of some IP applications over STANAG 5066.

U.1. Changes in This Edition

The core text of this annex is taken from Annex F Section F.12 of Edition 3.

The functional differences between this specification and Edition 3 are set out in Section U.9.

The key functional changes are:

1. Support of IPv6
2. Significant clarification and tidiness.

U.2. Model of Operation

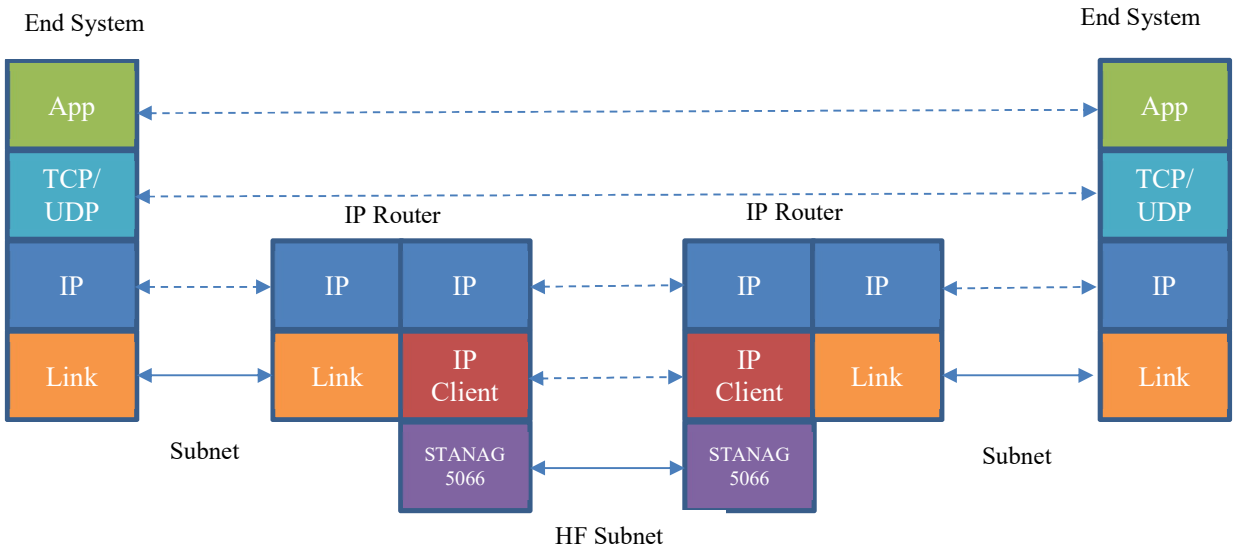


Figure U-1: IP Client HF Subnet Model

The general model of IP usage and IP Client is shown in Figure U-1. Applications

communicate end to end, over end to end layer protocols such as TCP and UDP. IP is used to communicate over a series of one or more subnets. IP is a per-hop protocol with end to end implications. For any subnet, a link protocol appropriate to subnet technology is used to communicate over the subnet. IP Client, specified in this annex is used to communicate over an HF subnet using STANAG 5066. IP Client is a simple layer protocol to enable IP to be exchanged using STANAG 5066.

In most configurations, the IP Client will be logically associated with an IP Router, and not an application, end system or host. It is possible to provide a router function with IP Client co-resident on an end system.

It is also possible to configure and end system to directly use IP Client, and for the end system to have an IP address on the subnet.

U.3. Scope of Application

The model shown in Figure U-1 is general purpose, and in principle can be used to support any IP service running over an HF subnet using IP Client. In practice, the choice of IP service that can be usefully deployed over IP Client is limited. Under load, the interaction between TCP and an IP Client subnet is inefficient in many configurations, and for many deployments use of IP Client is not suitable for TCP and HTTP provision.

IP Client is suitable to provide a range of IP services, such as ICMP Ping, and some low volume protocols operating over UDP.

U.4. General Requirements

An IP client implementation **shall** be capable of sending and receiving encapsulated IP datagrams with unicast (i.e., point-to-point) IP addresses, using both ARQ- and non-ARQ-transmission modes in STANAG 5066.

An IP client **may** be capable of sending and receiving encapsulated IP datagrams with multicast (i.e., point-to-multipoint) IP addresses, using non-ARQ-transmission modes.

U.5. Encapsulation of the IP Datagram using STANAG 5066 Service

IP datagrams **shall**(1) be encapsulated within the U_DPU field of S_UNIDATA_REQUEST Service specified in Annex A, and delivered to clients at the destination node(s) within the U_DPU field of S_UNIDATA_INDICATION Service. There are no framing characters required or allowed.

The first byte of the header of the IP datagram **shall**(2) be aligned with the first byte of the U_PDU field within the primitive, and so on to the last byte of the IP datagram

and U_PDU field.

The encoded bytes of an IP datagram submitted for transmission over the subnetwork **shall**(3) be bit-/byte- aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_REQUEST primitive.

The least-significant bit (LSB) of each octet in the IP datagram **shall**(4) be aligned with the LSB of the U_PDU's octet.

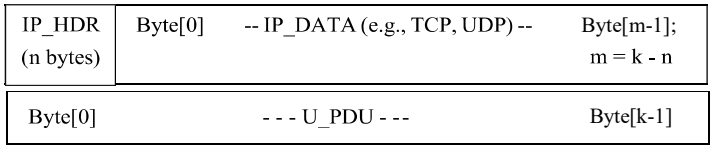


Figure U-2: - Mapping of IP Datagram into an S_UNIDATA_REQUEST U_PDU

U.5.1. IPv4 and IPv6

Both IPv4 and IPv6 datagrams can be carried. The version of IP used can be determined from the first four bits of the first byte of the IP Header. Choice of IP version is transparent to this protocol, but IP Client implementations **may** need to control actions based on IP version. If needed, this version is straightforward to determine.

U.6. IP-Client Subnetwork Service Requirements

IP clients **shall**(1) bind to the HF Subnetwork at SAP ID 9.

IP support by the HF subnetwork **shall**(2) be configurable to use either ARQ or non-ARQ delivery services.

Selection of the delivery mode (established in the S_UNIDATA_REQUEST message), and priority should be chosen based on the type of IP address and other information in the IP datagram, such as the IP protocol and protocol-specific features such as port.

An IP client **may** set the subnetwork's default service requirements in the S_BIND_REQUEST message as a function of the most likely traffic that it expects to process.

An IP client **shall**(4) be capable of overriding the subnetwork's default Service Type requirements and dynamically setting the S_UNIDATA_REQUEST Delivery Mode for

each IP Datagram submitted to the HF subnetwork.

U.6.1. IP Support using ARQ Service

HF subnetwork support using reliable point-to-point delivery between a pair of nodes is preferred for efficiency in the IP and higher-layer protocols, but in general cannot support IP-multicast protocols. [NB: The exceptional case is when an IP multicast address can be mapped to a STANAG 5066 unicast address, e.g., when tunnelling multicast traffic over an HF point-to-point link.]

The service definition for reliable-IP datagram delivery using the ARQ service **shall**(1) be as follows:

1. Transmission Mode = ARQ,
2. Delivery Confirmation = NODE DELIVERY,
3. Deliver in Order = IN-ORDER DELIVERY or AS-THEY-ARRIVE
4. Priority set according to Quality of Service as described in Section U.8.

U.6.2. IP Support using non-ARQ Service

If IP-multicast address groups are supported within the HF subnetwork environment (i.e., an application wishes to take advantage of the broadcast nature of the HF channel to support IP multicast), then the HF subnetwork **shall**(1) be configured in non-ARQ mode to support this requirement.

IP support using non-ARQ service **may** be used for IP unicast services.

For IP datagrams using non-ARQ service, the HF subnetwork service **shall**(2) be configured as follows:

1. Transmission Mode = non-ARQ,
2. Delivery Confirmation = none,
3. Deliver in Order = AS-THEY-ARRIVE.
4. Priority set according to Quality of Service as described in Section U.8.

The number of repeats for the D_DPUs in the service **may** be set to a value greater than one to provide some increased probability of receipt and reliability when using the subnetwork for IP multicast support.

U.6.3. Addressing using Independent Subnets (Router Architecture)

The IP Client Architecture, as shown in Figure U-1, interconnects IP subnetworks.

This is anticipated to be the most common mode of deployment. This means that the IP addressing is at a higher level than STANAG 5066 addressing. Two models of managing the address mapping are described. An IP Client implementation **may** implement one or both modes. IP users will be connected to independent subnets, which will access the IP Client through a router.

An IP Client product has two basic choices to support this:

1. It can operate by connecting to an IP router, using an implementation chosen communication mechanism.
2. It can provide IP router functionality as part of the IP Client product.

An IP Client implementation will be associated with an IP router on the local subnet. Communication between IP Client and the IP router is an implementation choice.

There are two choices for determining STANAG 5066 addresses to which an IP packet should be sent using IP Client.

U.6.3.1. Subnet based mapping of STANAG 5066 Addresses

The destination STANAG 5066 address is determined from the IP Subnet of the IP address, with an optional default mapping. This simple approach can be used in support of fixed IP routing, as it will be possible to specify a fixed mapping.

U.6.3.2. Link based mapping of STANAG 5066 Address

A second approach is to associate IP Client with a link to a remote router, and to associate the STANAG 5066 address of the peer with this link. This approach is preferable when dynamic IP routing is used between a pair of routers connected over HF, as it allows the mapping to STANAG 5066 addresses to not be affected by the IP routing.

U.6.4. Addressing using a Single Subnet (Bridged Architecture)

Another approach is to assign IP addresses on either side of an HF link to the same IP subnet. This leads to IP Client operating with a bridged architecture. In this mode of operation, IP client will need to operate as a bridge and map between the whole IP address and STANAG 5066 Address. This mode of operation will generally be impractical, but may be appropriate to support HF systems with a single node that does not have other IP connectivity.

U.6.5. IP Datagram Queueing

A key choice for IP Client implementation is handling STANAG 5066 flow control, when the load of arriving IP packets is greater than the HF subnet can immediately handle. A simple choice is to drop the IP packet when there is STANAG 5066 flow control. This is a simple approach, which follows the IP model of discarding packets on

congestion and expecting the higher layers to adapt. Because of long HF delays, this discarding can lead to significant inefficiency, particularly when ARQ is used.

The alternate option is to queue arriving IP packets. This can lead to improved performance in some situations, but large queues building up with very long delays will lead to a different type of performance issues. Handling this is a key implementation choice.

U.7. Router Functions

There is a close relationship between IP Client and an associated router. Consideration needs to be taken as to where some functions are provided.

U.7.1. Segmentation and Reassembly

Any Unidata containing IP data **shall**(1) contain a complete IP datagram.

Segmentation and re-assembly, if used will generally be performed by the router. Segmentation and re-assembly **may** be performed by IP Client.

U.7.2. Internetwork Control Message Protocol

The IP Client and Router **shall** jointly ensure that various standard requirements are addressed between them:

1. That the Internet Path MTU discovery (PMTU) Protocol [RFC1191] for IP datagrams marked with the DON'T_FRAGMENT flag that also exceed the HF subnetwork MTU size.

2. That Internetwork Control Message Protocol (ICMP) is supported to provide MTU discovery.

U.8. Selecting Options to Provide Best Quality of Service

For each IP packet handled, IP Client can make a number of choices:

1. ARQ vs Non-ARQ.
2. Priority.
3. Queueing strategy, as described in Section U.6.5.
4. Discard/Filtering. IP Client may choose to discard certain IP packets.

Considerations for this choice can include general conditions, such as typical transfer speed and SNR. The most useful choices are based on each IP packet. Options that **may** be considered include:

1. Destination Address.
2. IP Protocol (e.g., ICMP, GRE, UDP, TCP etc)
3. For IP protocols with Ports, in particular TCP and UDP, the port used.

It is **recommended** to have this choice configurable. The following considerations may be helpful.

1. ARQ generally gives better performance than non-ARQ.
2. Some queuing generally leads to better performance, but very long queues need to be avoided.
3. Discarding traffic that is not explicitly allowed can be helpful.
4. Giving higher priority to control traffic such as ICMP is generally desirable.

U.9. Changes in Edition 4

General descriptive text is added, including notes on recommended scope of application of IP Client.

This annex is changed from mandatory to optional.

Support for IPv6 is added. This needed clarification of procedure, but no change of protocol.

Support of multicast is made optional. This can be non-trivial to support, and its benefit is unclear.

Removal of controls related to Differentiated Services and TOS. These are not relevant to modern IP networking and added unnecessary complexity.

Address handling rewritten, as Edition 3 text does not make practical sense. Addressing needs to be handled as subnet level.

ANNEX V COMPRESSED FILE TRANSFER PROTOCOL (Optional)

Compressed File Transfer Protocol (CFTP) provides a protocol for transferring basic SMTP messages over STANAG 5066. It replaces the earlier HMTP protocol, which did not provide compression.

The Compressed File Transfer Protocol (CFTP) **may** be used to meet requirements in the NATO C3 Technical Architecture and the NATO Common Standards Profile (NCSP) for use of the SMTP Protocol. CFTP is used to reliably send compressed SMTP e-mail over a STANAG 5066 HF subnetwork from one message server to another.

CFTP is a vendor-defined standard derived from client definitions made in earlier, non-mandatory, versions of STANAG 5066 Annex F.

V.1. Changes in This Edition

The text of this annex is taken from Annex F Section F.14 of Edition 3.

There are no changes to this text.

V.2. General Requirements

Implementations of STANAG 5066 **may** provide a CFTP client. If provided, a CFTP client **shall**(1) conform to the requirements specified herein.

The CFTP protocol **shall not** be used for Formal or High Grade Military Messaging (i.a. military orders). For Formal or High Grade Military Messaging, ACP 142 as specified in Annex Q **shall**(2) be used. The CFTP protocol **may** be used for informal interpersonal e-mail only.

CFTP **shall**(3) operate within the node model shown, providing transfer services from one SMTP e-mail server to another via the CFTP client.

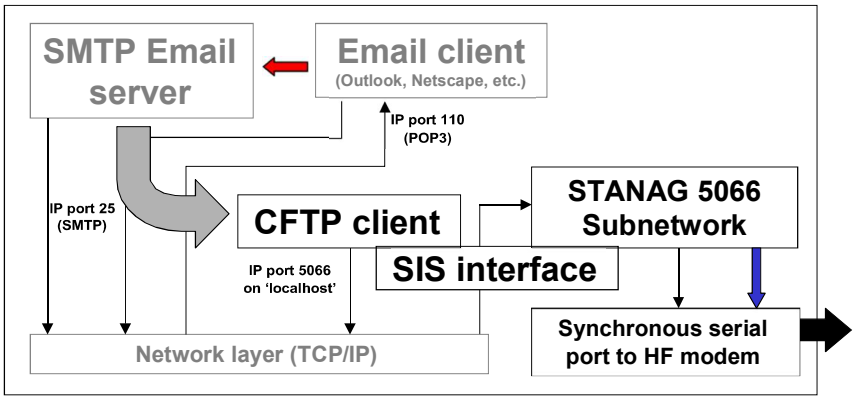


Figure V-1: CFTP Client and Node model

In general, the interaction of the CFTP client with the mail-server is beyond the scope of this STANAG. In operation, when an email message is received at a 5066 node, it is placed in an incoming mail folder (mail spool directory). The CFTP client, also called the Delivery Agent (DA), removes mail from this incoming folder and processes the mail for delivery over HF via 5066. The CFTP DA compresses the message and information about the message, e.g. size, id, recipients, etc. into a file. This compressed file is then transferred to the destination 5066 node(s) using the original form of the Basic File Transfer Protocol (BFTP), referred to here as BFTPv1. The original BFTPv1 format is incorporated directly in the CFTP specification below to free it from dependencies on (and incompatibilities with) BFTP specification found in STANAG 5066 Ed3.

V.3. CFTP Subnetwork Service Requirements

CFTP clients **shall** bind to the HF Subnetwork at SAP ID 12.

V.4. CFTP Connection-Oriented Protocol

The CFTP application **shall** use the earlier form of the RCOP Protocol Data Unit ("RCOPv1") defined in the Figure below. [NB: As a historical note, this corresponds to the definitions of the RCOP protocol found in the original information-only Annex F Edition 1. It does not include the Application Identifier within the PDU Header that has been added in STANAG 5066 Edition 2.]

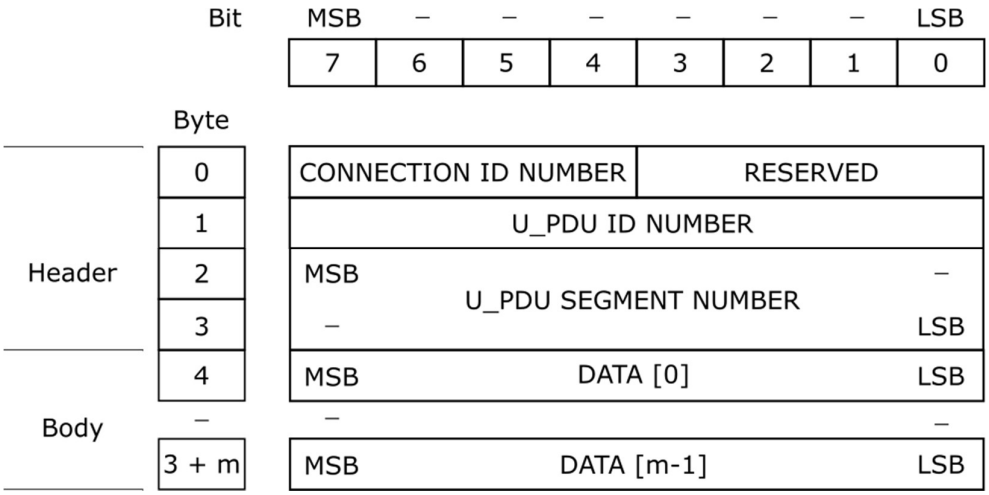


Figure V-2:. CFTP Protocol Data Units (identical in format to (Original) RCOP Protocol Data Units (RCOPv1) from STANAG 5066 Annex F Edition 1)

The following are required for RCOPv1 PDUs:

1. The connection ID number **shall** be a value from 0-15. Connection ID number 0 shall be reserved for non-multiplexed connections.
2. The reserved bits **shall** be set to 0.

3. The U_PDU ID numbers **shall** be assigned consecutively to U_PDU (i.e., files).
4. The U_PDU segment number **shall** be assigned consecutively to segments within a single U_PDU. The first segment transmitted **shall** be assigned segment number 0. If a U_PDU is not segmented, the single segment transmitted **shall** be assigned number 0.

V.4.1. Compressed File-Delivery and Delivery-Confirmation

Compressed files **shall**(1) be transferred from one CFTP client to another using the Edition 1 Basic File Transfer Protocol ('BFTPv1') as defined in the subsections below. Client Delivery confirmation **shall**(2) be provided using the CFTP Message Acknowledgement, defined in the subsequent section (see Section V.4.3), as the body-part of the PDU.

In principle, up to 256 files could be transferred concurrently using the unique RCOPv1 U_PDU ID number for each transfer, using the U_PDU_ID as the identifier to match acknowledgements to the file acknowledged. As there is no negotiation protocol currently defined to determine if a given receiving node supports this capability, a sending node **must** have prior knowledge that a given receiving node supports concurrent multiple- file delivery.

Consequently, the Client Delivery confirmation protocol is nominally stop-and-wait — a new file **should not**(1) be sent with a given U_PDU ID until a message acknowledgement has been received. However, this recommendation **may**(1) be relaxed to allow concurrent multiple-file delivery when the sending node has prior knowledge that the receiving node supports the capability. New implementations of CFTP **should** support concurrent multiple file delivery.

V.4.1.1.BFTPv1 Specification [NB: corresponding to the original Edition 1 BFTP specification]

The format for the basic-file-transfer-protocol data unit Version 1 (BFTPv1) **shall**(1) be in accordance with the following Figure, which defines a header part and a file-data part for the BFTP_PDUv1.

BFTP_HDR	Byte[0]	- - - FILE_DATA[] - - -	Byte[p-1]

Figure V-3:: Basic FTP Version 1 Protocol Data Unit (BFTPv1 PDU)

The detailed structure of the BFTPv1_PDU **shall**(2) be in accordance with the following Figure, and provide the following information fields:

1. BFTPv1_PDU Header Part:
    - SYNCHRONIZATION - two bytes corresponding to the control bytes DLE (Data Link Escape) and STX (Start of Text).

- SIZE_OF_FILENAME - one octet in size.
- FILE_NAME - a variable length field, equal in size to the value specified by the SIZE_OF_FILENAME field.
- SIZE_OF_FILE - a four-octet field.

2. BFTPv1_PDU Body Part:
   - FILE_DATA[] - a variable length field, equal in size to the value specified by the SIZE_OF_FILE field.

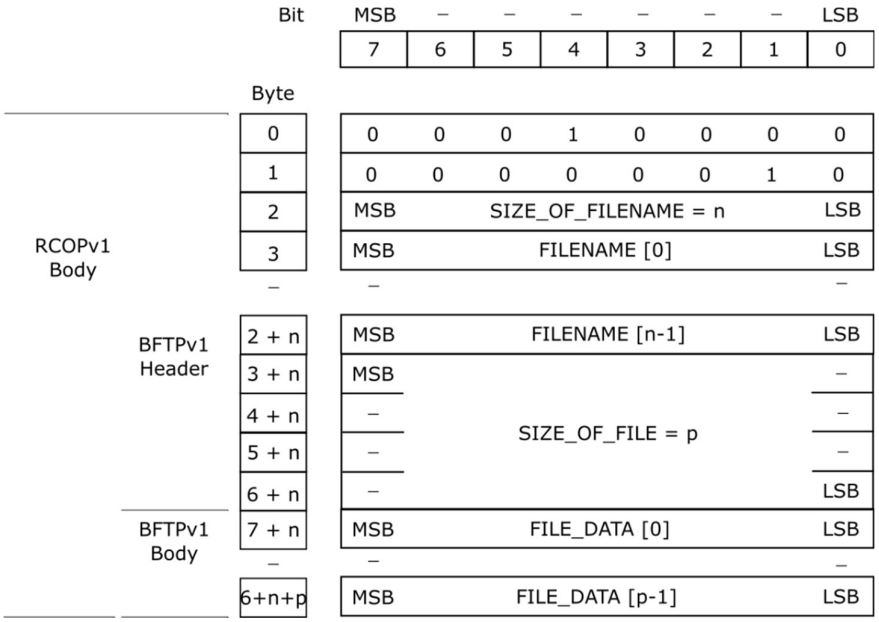


Figure V-4: BFTPv1 Protocol Data Unit Structure

The SIZE_OF_FILENAME field **shall**(1) be a 1-octet fixed-length field whose value (n) **shall**(2) equal the number of octets used to encode the FILENAME field.

The FILENAME field is **shall**(1) be a variable-length field, the size of which **shall**(2) be specified by the value (n) of the field SIZE_OF_FILENAME. This field represents the name of the file sent using the Basic File Transfer Protocol. The first byte of the filename **shall**(3) be placed in the first byte of this field, with the remaining bytes placed in order. The semantics of file names and naming conventions are beyond the scope of this STANAG (e.g., there is no requirement that the filename be a null-terminated character string.)

The SIZE_OF_FILE field **shall**(1) be a 4-octet fixed-length field whose value **shall**(2) specify the size (p) in octets of the file to be sent. The first octet of the SIZE_OF_FILE field **shall**(3) be the highest order byte and the last byte the lowest order byte of the field's binary value.

V.4.2. **BFTPv1 Segmentation and Reassembly Requirements**

If the BFTPv1_PDU exceeds the maximum size of the data field permitted in the RCOPv1 PDU (i.e, if the CTFP_PDU is larger than the MTU_size less 4 octets (i.e., MTU-4)), the CFTP client **shall** segment the BFTPv1 PDU, placing successive segments in RCOPv1 PDUs (original Edition 1 format) with consecutive U_PDU sequence numbers.

When received, the CFTP client **shall**(2) reassemble the BFTPv1 PDU if it determines that the BFTPv1 PDU has been segmented. Subject to local-host file naming conventions, the CFTP client **shall**(2) store the received file with the name transmitted in the header with the file. [*NB: there is no guarantee therefore that the file will be stored on the destination host with the same name that it was sent.]*

V.4.3. **Message Acknowledgement**

Client Delivery confirmation **shall**(1) be provided using the Message Acknowledgement defined below, sent as the body-part of a CFTP/RCOPv1 PDU.

	Bit	MSB 7	6	5	4	3	2	1	LSB 0
	Byte								
BFTPv1 ACK	0	0	0	0	1	0	0	0	0
(RCOPv1 body part)	1	0	0	0	0	1	0	1	1

Figure V-5: BFTPv1 Message Acknowledgement Structure

On receiving the last byte of the message, the receiving client **shall**(2) send the Message Acknowledgement (0x10 0x0B) — with the same RCOPv1 U_PDU_ID NUMBER and CONNECTION ID NUMBER as the message being acknowledged — to confirm that the entire message has been received. (N.B. This is equivalent to the "ZEOF" message of the Z-modem protocol.)

V.5. CFTP Compression/Decompression

The compressed file **shall** be created and decompressed in accordance with RFCs 1950, 1951 and 1952 (i.e., the gzip utility defined in RFC1952).

V.6. CFTP Compressed File Data Format

The compressed file data **shall** be formatted as a series of fields. The fields are separated by the linefeed <LF> character 0x0A. The fields and the order in which they are compressed are described in the following table.

Table V-1 CFTP Mail File Structure

Order of Compression	Field Name	Description
1	MessageID	The MessageID field is represented by an arbitrary string that serves as the ID for the message. The MessageID is unique to an e-mail message. It is not the same as the ID in the email message that follows "Message-ID:" in the header. When the compressed file is decompressed, the MessageID is used as the root filename for the decompressed components. The MessageID must be less than 256 characters and is composed of upper/lowercase alphanumeric
2	RecipientList	The RecipientList is a string containing e-mail addresses extracted from the e-mail message, each address delimited by the "," character (0x2c). The first address in the recipients list is the "Return-Path". There can be cases where there is no return path, e.g. the mail is being bounced by a Mail Transfer Agent. In these cases, the first address will be an empty string (i.e., either a single space [0x20] character or no characters at all) and it will be followed by a comma
3	MessageSize	The MessageSize is encoded as a decimal number in string format. It represents the size (in bytes) of the Message field that follows the MessageSize field.
4	Message	Actual message as received by an SMTP receiver ***i.e. including the terminating sequence <CRLF>.<CRLF> and any additional characters that may be required for transparency as defined by***

Note 1. All characters are 8 bits.
Note 2. The terminating sequence **<CRLF>.<CRLF>** is that shown in Example 1 of RFC 821 and equates to the 5 ASCII Characters with codes, in hexadecimal, of 0x0D, 0x0A, 0x2E, 0x0D, 0x0A.

V.7. Detailed Description of CFTP

1) An e-mail client is used to send an e-mail to an SMTP server.
2) The CFTP application extracts the e-mail message from the directory in which it was placed by the SMTP server. An example e-mail message in the correct format is shown in Figure F-23 below.
3) The CFTP mail-file shall be built as follows:

<MessageID><LF> // The MessageID field shall(1) be represented by a character string. The MessageID **shall**(2) be unique to an e-mail message. It is not the same as the ID in the email message that follows "Message-ID:" in the header. When the compressed file is decompressed, the MessageID **shall** be used as the root filename for the decompressed components. The MessageID **must** be less than 256 characters and **may** be composed of upper/lowercase alphanumeric

<RecipientsList><LF>	//	The RecipientList **shall**(1) be a character string containing e-mail addresses extracted from the e-mail message, each address(0x2c). The first address in the recipients list **shall**(2) be the "Return-Path". There **may** be cases where there is no return path, e.g. the mail is being bounced by a Mail Transfer Agent. In these cases, the first address **shall**(3) be an empty string (i.e., either a single space [0x20] character or no characters at all) followed by a comma (i.e, a "," character with octet value = 0x2c). The recipients list **must** be less than 10240 characters separated by "," character
<MessageSize><LF>	//	The MessageSize shall be encoded as a decimal number in string format terminated by the linefeed character. It represents the size (in bytes) of the Message field that follows.
<Message>	//	The Message field **shall** contain the e-mail message body part(s) extracted from the SMTP envelope.

4) The CFTP message (including header) shall be compressed in accordance with RFCs 1950, 1951 and 1952 using an application such as gzip.
5) The compressed CFTP message shall be encapsulated within a BFTPv1 PDU (i.e., it has a BFTPv1 header prepended to it, and the CFTP message shall be byte aligned within the FILE_DATA[] field of the BFTPv1 PDU.
6) The BFTPv1 message (i.e., BFTPv1 PDU) shall be segmented if necessary.
7) Each BFTPv1 PDU segment shall have an RCOPv1 header added (in accordance with Annex F.14.3.).
8) Each RCOPv1 packet shall be packaged into an S_UNIDATA_REQUEST and transferred using a Soft Link Data Exchange.
9) On reception the BFTPv1 message shall be reassembled, if required, and decompressed using a method compliant with RFC 1952 and the CFTP message reconstructed.
10) The received email messages shall be forwarded to an SMTP server using a standard SMTP dialogue based on information extracted from the CFTP header and inserting the "message" field into the payload of the SMTP message generated.

```
Received: from northampton (unverified [127.0.0.1]) by northampton.pdw<CRLF>
 (Rockliffe SMTPRA 4.2.4) with SMTP id
 <B0000000133@northampton.pdw> for
<root@essex.pdw>;<CRLF>
 Wed, 9 May 2001 12:09:03 +0100<CRLF>
Message-ID:
<001f01c0d878$74da90d0$0d02a8c0@pdw><CRLF>
From: "Northampton"
<administrator@northampton.two><CRLF> To:
<root@essex.pdw><CRLF>
Subject: Test <CRLF>
Date: Wed, 9 May 2001 12:09:03
+0100<CRLF> MIME-Version:
1.0<CRLF>
Content-Type:
        text/plain;<CRLF>
        charset="iso-
        8859-1"<CRLF>
Content-Transfer-Encoding:
7bit<CRLF> X-Priority:
3<CRLF>
X-MSMail-Priority: Normal<CRLF>
X-Mailer: Microsoft Outlook Express 5.50.4522.1200<CRLF>
X-MimeOLE: Produced By Microsoft MimeOLE V5.50.4522.1200<CRLF>
<CRLF>
This is the body of the test email<CRLF>
<CRLF>
.<CRLF>
```

Figure V-6: Example email in the Correct Format

The red and blue text above is the message body with text in blue being the terminating sequence:

<CRLF>.<CRLF> i.e 0x0D, 0x0A, 0x2E, 0x0D, 0x0A.

AComP-5066 (A)(1)