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.

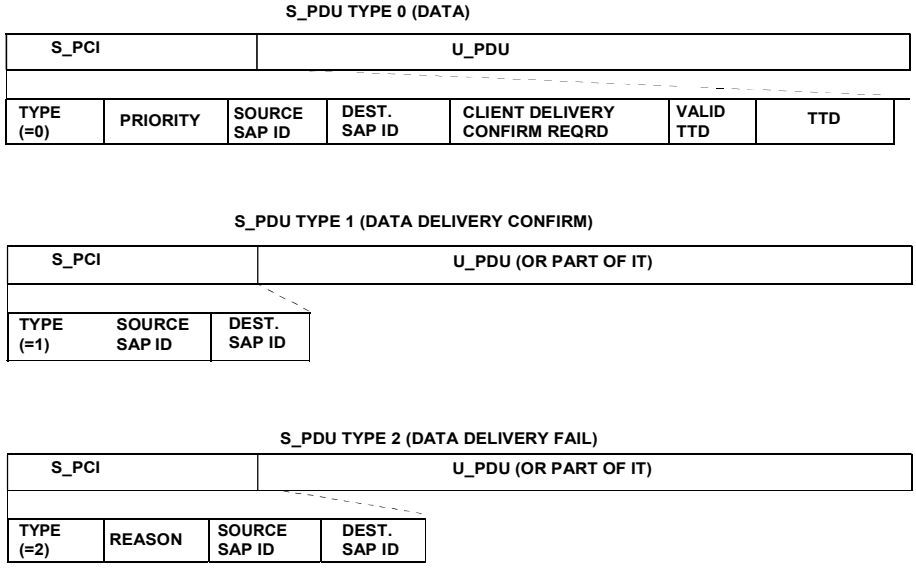

Figure A-1: Generic Encoding of S_PDUs

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 :

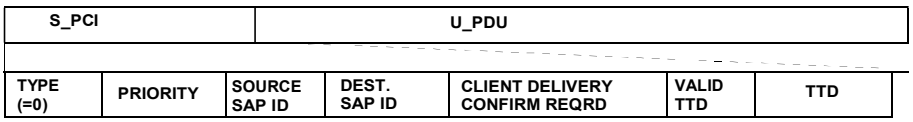

Figure A-2: Generic Encoding of the DATA S_PDU

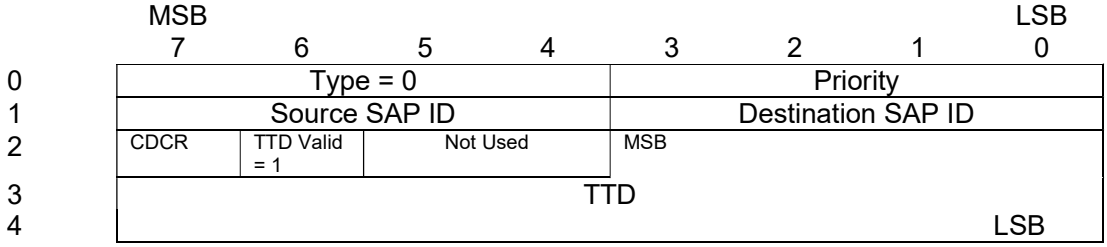

Figure A-3: Bit-Field Map of the DATA S_PDU S_PCI with valid TTD

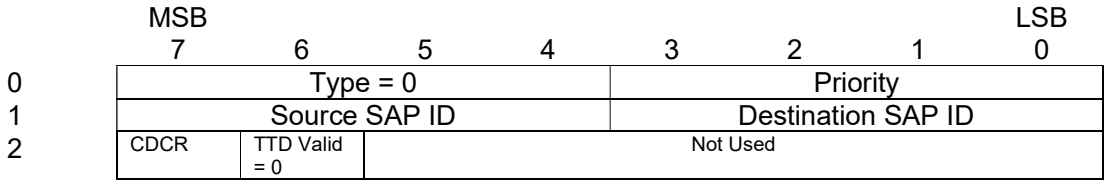

Figure A-4: Bit-Field Map of the DATA S_PDU S_PCI without TTD

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 :

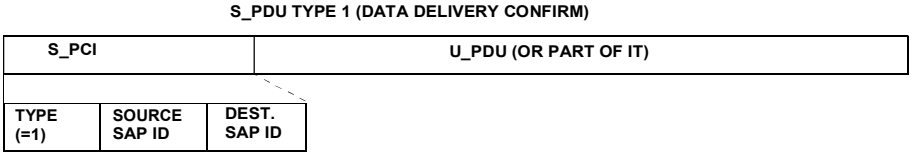

Figure A-5: Generic Encoding of the DATA DELIVERY CONFIRM S_PDU

	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 :

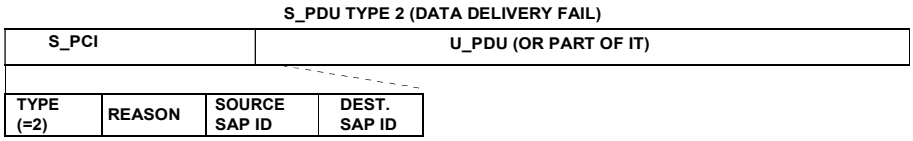

Figure A-7: Generic Encoding of the DATA DELIVERY FAIL S_PDU

	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.

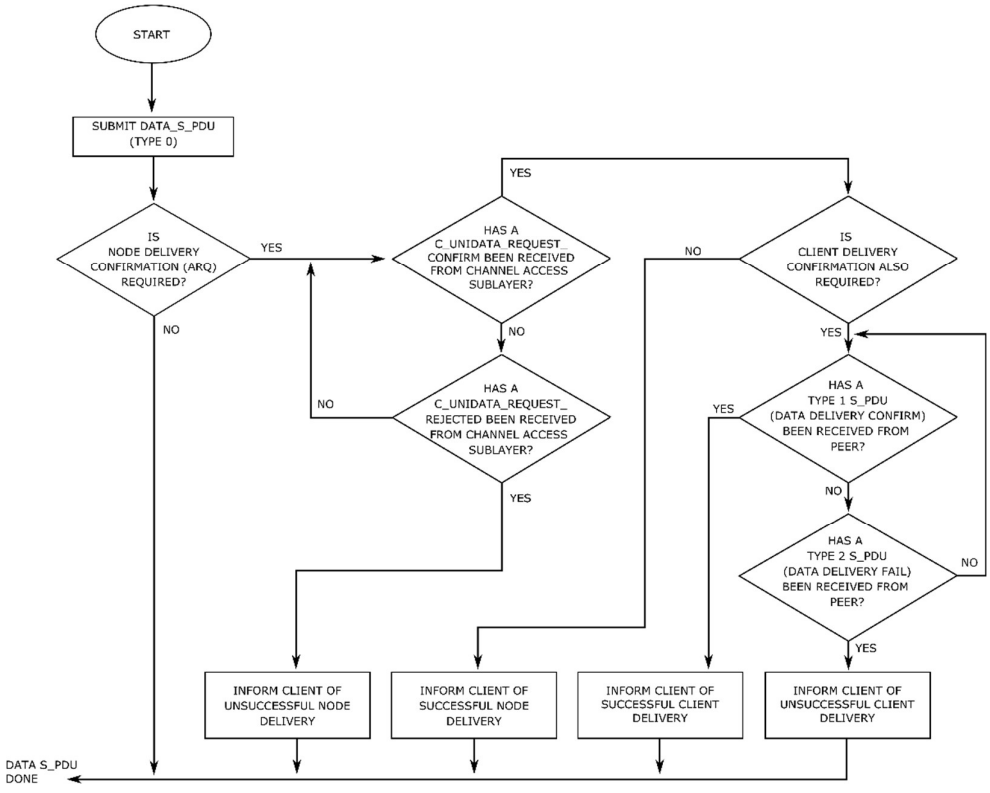

Figure A-9: Data Exchange Procedures: SENDING PEER

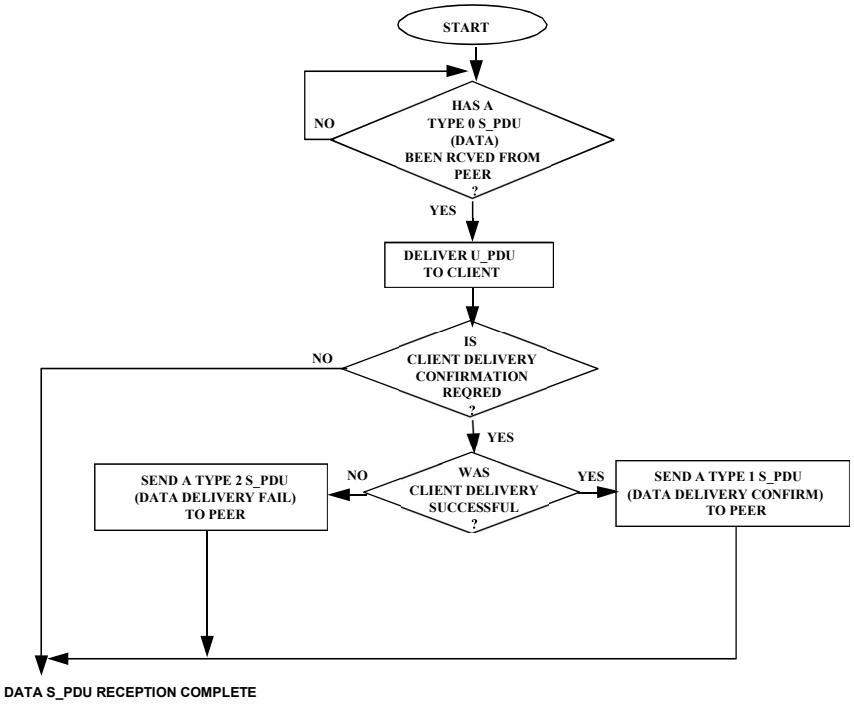

Figure A-10: Data Exchange Procedures: RECEIVING PEER

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.