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 client-server relationship governs the interaction between the HF subnetwork and the users of the subnetwork. 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.

Annex F contains a recommended definition of the various subnetwork clients. For the purposes of this STANAG, the subnetwork client definitions in Annex F are not mandatory. Data submitted by the clients to the Subnetwork Interface Sublayer must be in the form of primitives with the format as described in this document. Clients are responsible for segmenting larger messages into User Protocol Data Units (U_PDUs). A U_PDU format that supports this segmentation is defined in Annex F, but remains outside of the scope of the mandatory requirements on the client-to-subnetwork interface.

The Subnetwork Interface Sublayer treats all clients connected to it in the same manner irrespective of the application performed by these clients. The only distinguishing factor between clients is their **Rank** that is a measure of their importance. See Annex H.5 for further information on the rank of clients. Certain service requests made by higher ranked clients may take precedence over requests made by lower ranked clients.

A.1.1 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 four types of sessions:

1. Soft Link Data Exchange Session
2. Hard Link Data Exchange Session
3. Broadcast Data Exchange Session
4. Reserved

All sessions apart from the broadcast data exchange session require the making of a point-to-point physical link with a specified remote node.

[1] SAPs are equivalent to the “ports” of the TCP protocol.

A.1.1.1 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) and from which a client has not requested a Hard Link Data Exchange Session.

The Subnetwork Interface Sublayer **shall** (2) initiate Soft Link Data Exchange Sessions as needed, following the procedure described in Section A.3.2.1.1.

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.3.2.1.2.

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.

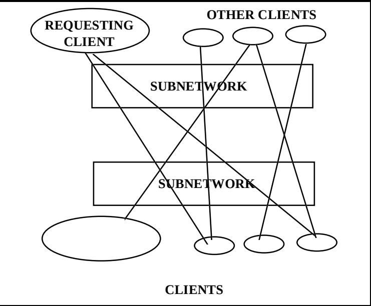
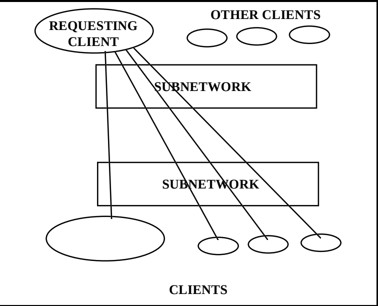
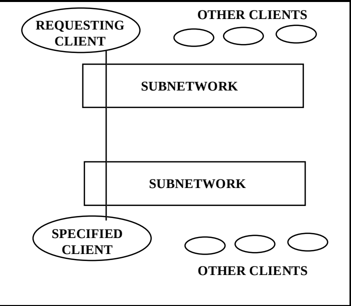
In order to provide “balanced” servicing of the queued U_PDUs, a Soft Link Data Exchange Session **shall** (5) not be maintained for a period which exceeds a specified maximum time if U_PDUs of appropriate priorities are queued for different node(s).

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.) must be chosen in the context of a particular configuration (i.e. size of network, etc).

A.1.1.2 Hard Link Data Exchange Sessions

The second type of data exchange session is the Hard Link Data Exchange Session. A Hard Link Data Exchange Session **shall** (1) be initiated at the explicit request of a client in accordance with the procedures for establishing and terminating hard link sessions specified in Sections A.3.2.2.1 and A.3.2.2.2.

A client may request a Hard Link Data Exchange Session in order to ensure that a physical link to a specified node is maintained (irrespective of the destinations of other queued U_PDUs) and optionally to partially or fully reserve the capacity of such a link. The three types of Hard Links that may be established are depicted below in the following figure:

HARD LINK TYPE 0 – Link Reservation	**HARD LINK TYPE 1- Partial-Bandwidth Reservation**	**HARD LINK TYPE 2 – Full-Bandwidth Reservation**
 REQUESTING CLIENT OTHER CLIENTS SUBNETWORK SUBNETWORK CLIENTS	 REQUESTING CLIENT OTHER CLIENTS SUBNETWORK SUBNETWORK CLIENTS	 REQUESTING CLIENT OTHER CLIENTS SUBNETWORK SUBNETWORK SPECIFIED CLIENT OTHER CLIENTS

Hard-Link Data-Exchange Session Types

A.1.1.2.1 Type-0 Hard Link: Physical Link Reservation

A Hard Link of Type-0, also called a Hard Link with Link Reservation, **shall** (1) maintain a physical link between two nodes.

The Type-0 Hard Link capacity **shall** (2) not be reserved for any given client on the two nodes.

Any client on nodes connected by a Hard Link of Type 0 **shall** (3) be permitted to exchange data over the Hard Link.

Any client on either node other than the client that requested the Hard Link **shall** (4) gain access to the link only as a Soft-Link Data Exchange Session and may lose the link when the originating client terminates its Hard Link Data Exchange Session.

A.1.1.2.2 Type-1 Hard Link: Partial-Bandwidth Reservation

A Hard Link of Type 1, also called a Hard Link with Partial Bandwidth Reservation, **shall** (1) maintain a physical link between two nodes.

The Type 1 Hard Link capacity **shall** (2) be reserved only for the client that requested the Type 1 Hard Link between the two nodes. The requesting client may send user data to any client on the remote node, and may receive user data from any client on the remote node only as a Soft-Link Data Exchange Session.

Clients that are not sending data to or receiving data from the client that requested the Type 1 Hard Link **shall** (3) be unable to use the Hard Link. Any client using the link may lose the link when the originating client terminates its Hard Link Session.

A.1.1.2.3 Type-2 Hard Link: Full-Bandwidth Reservation

A Hard Link of Type 2, also called a Hard Link with Full Bandwidth Reservation, **shall** (1) maintain a physical link between two nodes.

The Type 2 Hard Link capacity **shall** (2) be reserved only for the client that requested the Type 1 Hard Link and a specified remote client. No clients other than the requesting client and its specified remote client **shall** (3) exchange data on a Type-2 Hard Link.

A.1.1.3 Broadcast Data Exchange Session

The third 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 hard link requests or ARQ Data U_PDUs. Alternatively the Subnetwork Interface Sublayer can unilaterally initiate and terminate Broadcast Data Exchange Sessions.

A.2 Primitives Exchanged with Clients

Communication between the client and the Subnetwork Interface Sublayer uses 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 subnetwork clients. 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, Rank, SAP ID)	S_BIND_ACCEPTED (SAP ID, MTU)
	S_BIND_REJECTED (Reason)
S_UNBIND_REQUEST ( )	S_UNBIND_INDICATION (Reason)
S_HARD_LINK_ESTABLISH (Link Priority, Link Type, Remote Node Address, Remote SAP ID)	S_HARD_LINK_ESTABLISHED (Remote Node Status, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
	S_HARD_LINK_REJECTED (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
S_HARD_LINK_ACCEPT (Link Priority, Link Type, Remote Node Address, Remote SAP ID)	S_HARD_LINK_INDICATION (Remote Node Status, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
S_HARD_LINK_REJECT (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID)	
S_HARD_LINK_TERMINATE (Remote Node Address)	S_HARD_LINK_TERMINATED (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID)
	S_SUBNET_AVAILABILITY (Subnet Status, 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, U_PDU)
	S_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, U_PDU)
	S_UNIDATA_INDICATION (Source Node Address, Source SAP ID, Destination Node Address, Destination SAP ID, Priority, Transmission Mode, U_PDU)
S_EXPEDITED_UNIDATA_REQUEST (Destination Node Address, Destination SAP ID, TimeToLive, Delivery Mode, U_PDU)	S_EXPEDITED_UNIDATA_REQUEST_CONFIRM (Destination Node Address, Destination SAP ID, U_PDU)
	S_ EXPEDITED_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, U_PDU)
	S_ EXPEDITED_UNIDATA_INDICATION (Source Node Address, Source SAP ID, Destination Node Address, Destination SAP ID, Transmission Mode, U_PDU)
	S_DATA_FLOW_ON( )
	S_DATA_FLOW_OFF ( )
S_MANAGEMNT _MSG_REQUEST (MSG)	S_MANAGEMENT_MSG_INDICATION (MSG)
S_KEEP_ALIVE ( )	S_KEEP_ALIVE ( )

A.2.1 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.2.1.1 S_BIND_REQUEST Primitive

Name :

S_BIND_REQUEST ( )

Arguments :

1. SAP ID,
2. RANK,
3. 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 second argument of this primitive **shall** (4) be "*Rank*". This is a measure of the importance of a client; the subnetwork uses a client's rank to allocate resources. A description of the use of the Rank argument may be found in Annex H and [1]. The range of values for the rank argument **shall** (5) be from 0 to 15. Clients that are not authorised to make changes to a node or subnetwork configuration **shall** (6) not bind with rank of 15.

The last argument of this primitive **shall** (7) be "*Service Type*" and identifies the default type of service requested by the client. 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. The "*Service Type*" argument is a complex argument and has a number of attributes that are encoded as specified in Section A.2.2.3.

A.2.1.2 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.

The Subnetwork Interface Sublayer can sense whether a client is physically disconnected in order to unilaterally declare this client as off-line; the S_KEEP_ALIVE primitive specified in Section A.2.1.17 provides this capability, though other implementation-dependent methods may be used in addition to this primitive.

[Note: The omission of SAP ID as an argument in this and other primitives implies a requirement on the stack supporting this connection to associate a SAP ID with a lower level connection (i.e., socket) and maintain this association.]

A.2.1.3 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.2.1.4 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 binary representation of the value in the table **shall** (4) be encoded in the Reason field of the primitive by placing the LSB of the value into the LSB of the encoded field for the primitive as specified in Section A.2.2.

A.2.1.5 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. As an implementation dependent example, if a new "High Ranked" client submits a S_BIND_REQUEST to come on-line but not enough resources are available, the Subnetwork Interface Sublayer may unilaterally declare a "Lower Ranked" client off-line. In such a case, the sublayer will send the lower-ranked client an S_UNBIND_INDICATION in order to release resources for the Higher-Ranked client.

The *Reason* argument of the S_UNBIND_INDICATION primitive **shall** (2) specify why the client was declared off-line. The binary representation of the value in the table **shall**(3) be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the encoded field for the primitive as specified in section A.2.2.

Reason	Value
Connection pre-empted by higher ranked client	1
Inactivity (failure to respond to "Keep alive")	2
Too many invalid primitives	3
Too many expedited data request primitives	4
ARQ Mode Unsupportable during Broadcast Session	5

A.2.1.6 S_UNIDATA_REQUEST Primitive

Name :

S_UNIDATA_REQUEST( )

Arguments :

1. Priority
2. Destination SAP ID
3. Destination Node Address
4. Delivery Mode
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. 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* **shall** (5) be a complex argument with a number of attributes, as specified by the encoding rules of Section A.2.2.28.2. This argument can be given the value of "DEFAULT" which means that the delivery mode associated with the U_PDU will be the delivery mode 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 the default delivery mode for this U_PDU.

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 final 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, e.g. GMT. 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.2.1.7 S_UNIDATA_REQUEST_CONFIRM Primitive

Name :

S_UNIDATA_REQUEST_CONFIRM

Arguments :

1. Destination Node Address
2. Destination SAP ID
3. 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** (3) 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_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 **shall** (7) 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** (8) 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** (9) be a configurable parameter in the implementation.

A.2.1.8 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

The binary representation of the value in the table **shall** (5) be mapped into the Reason argument of the primitive by placing the LSB of the value into the LSB of the encoded argument for the primitive as specified in section A.2.2

The *Destination Node Address* argument in the S_UNIDATA_REQUEST_REJECTED Primitive **shall** (6) 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.2.1.9 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 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, Non-ARQ w/ Errors, etc.

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:

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.2.1.10 S_EXPEDITED_UNIDATA_REQUEST Primitive

Name :

S_EXPEDITED_UNIDATA_REQUEST

Arguments :

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

Direction :

Client->Subnet Interface

Description :

The S_EXPEDITED_UNIDATA_REQUEST primitive **shall** (1) be used to submit a U_PDU to the HF Subnetwork for Expedited Delivery to a receiving client.

The argument *Destination SAP ID* **shall** (2) 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** (3) specify the HF subnetwork address of the physical HF node to which the receiving client is bound.

The argument *Delivery Mode* **shall** (4) be a complex argument with a number of attributes, as specified by the encoding rules of Section A.2.2.28.2. This argument can be given the value of “Default” which means that the delivery mode associated with the U_PDU will be the delivery mode specified by the client during “binding” (*Service Type* argument of S_BIND_REQUEST). The other values of the *Delivery Mode* can be used to override the default delivery mode for this U_PDU.

The argument *TimeToLive (TTL)* **shall** (5) specify the maximum amount of time the submitted U_PDU is allowed to stay in the HF Subnetwork before it is delivered to its final destination. If the TTL is exceeded the U_PDU **shall** (6) be discarded. A TTL value of 0 **shall** (7) define an infinite TTL**,** i.e. the subnetwork should try *forever* to deliver the U_PDU.

As soon as the Subnetwork Interface Sublayer accepts a S_EXPEDITED_UNIDATA_REQUEST primitive, it **shall** (8) immediately calculate its

TimeToDie (TTD) by adding the specified TTL (or the default maximum TTL value if the specified TTL is equal to 0) to the current Time of Day, e.g. GMT. The TTD attribute of a U_PDU **shall** (9) 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** (10) be the size of the U_PDU that is included in this S_UNIDATA_REQUEST Primitive.

The final argument, *U_PDU*, **shall** (11) be the actual User Data Unit (U_PDU) submitted by the client to the HF Subnetwork for expedited delivery service.

[Note: There is no *Priority* argument in the S_EXPEDITED_UNIDATA_REQUEST primitive. Although seemingly equivalent, there are a important differences between a S_UNIDATA_REQUEST primitive of the highest priority and a S_EXPEDITED_UNIDATA_REQUEST primitive. S_UNIDATA_REQUEST primitives of all priority levels are processed according to a set of rules that apply to Normal Data. U_PDUs submitted using S_EXPEDITED_UNIDATA_REQUEST primitives are treated differently, e.g. expedited U_PDUs should be queued separately from normal U_PDUs. When an expedited U_PDU is received, the transmission of normal data is halted and the expedited data is transmitted. When the expedited data has been sent the transmission of normal data is resumed again.]

The 5066 node management **shall** (3) track the number of S_EXPEDITED_UNIDATA_REQUEST primitives submitted by various clients. If the number of S_EXPEDITED_UNIDATA_REQUEST primitives for any client exceeds a configurable, implementation dependent parameter, node management **shall** (4) unilaterally disconnect the client using a S_UNBIND_INDICATION primitive with REASON = 4 = “Too many expedited-data request primitives”.

A.2.1.11 S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive

Name :

S_EXPEDITED_UNIDATA_REQUEST_CONFIRM

Arguments :

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

Direction :

Subnetwork Interface->Client

Description :

The S_EXPEDITED_UNIDATA_REQUEST_CONFIRM primitive **shall** (1) be issued by the Subnetwork Interface Sublayer to acknowledge the successful delivery of a S_EXPEDITED_UNIDATA_REQUEST primitive.

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

The *Destination Node Address* argument in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive **shall** (3) have the same meaning and be equal in value to the *Destination Node Address* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive is the response.

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

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

Just as specified for the S_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* argument in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM 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** (6) 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** (7) be a configurable parameter in the implementation.

A.2.1.12 S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive

Name :

S_EXPEDITED_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_EXPEDITED_UNIDATA_REQUEST_REJECTED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer to inform a client that a S_EXPEDITED_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), or if a U_PDU larger than the MTU is submitted.

The argument *Reason* **shall** (3) specify why the delivery failed with values defined for this field as specified 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

The binary representation of the value in the table **shall** (4) be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the encoded field for the primitive as specified in section A.2.2.1.

The *Destination Node Address* argument in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive **shall** (5) have the same meaning and be equal in value to the *Destination Node Address* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Destination SAP_ID* argument in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive **shall** (6) have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Size of Rejected U_PDU* argument **shall** (7) be the size of the U_PDU or part that is included in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive.

Just as specified for the S_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* argument in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM 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** (8) 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** (9) be a configurable parameter in the implementation.

A.2.1.13 S_EXPEDITED_UNIDATA_INDICATION Primitive

Name :

S_EXPEDITED_UNIDATA_INDICATION

Arguments :

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

Direction :

Subnetwork Interface->Client

Description :

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

The *Destination SAP ID* argument **shall** (2) be the SAP ID of the client to which this primitive is delivered.

The *Destination Node Address* argument **shall** (3) 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** (4) 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, Non-ARQ w/ Errors, etc.

The *Source SAP ID* **shall** (5) be SAP ID of the client that sent the U_PDU.

The *Source Node Address* **shall** (6) represent the node address of the client that sent the U_PDU.

The *Size of U_PDU* argument **shall** (7) be the size of the U_PDU that was sent and delivered in this S_UNIDATA_INDICATION S_Primitive.

The following four arguments **shall** (8) 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:

a) The *Number of Blocks in Error* argument **shall** (9) 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** (10) specify the number of ordered pairs in the *Array of Block-Error Pointers* argument.

b) The *Array of Block-Error Pointers* argument **shall** (11) 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** (12) equal the number of data blocks missing from the U_PDU because they were not received. This argument **shall** (13) specify the number of ordered pairs in the *Array of Non-Received-Block Pointers* argument.

d) The *Array of Non-Received-Block Pointers* **shall** (14) 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** (15) contain the actual received user data for delivery to the client.

A.2.1.14 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 S_FLOW_DATA_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_FLOW_DATA_ON primitive.

Depending on the implementation, the physical connection between the client(s) and the Subnetwork Interface Sublayer may provide an implicit flow-control mechanism that would make the use of these primitives unnecessary. For example, if the connection is implemented as TCP/IP Berkeley Sockets, the implicit flow-control mechanism of the TCP protocol may be utilized in which case these two primitives are redundant.

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.

A.2.1.15 S_MANAGEMENT_MSG_REQUEST Primitive

Name :

S_MANAGEMENT_MSG_REQUEST

Arguments :

1. MSG

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.

The subnetwork **shall** (3) accept this primitive only from clients which have bound with a rank of 15.

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 or EXPEDITED UNIDATA primitives defined elsewhere in this annex.

A.2.1.16 S_MANAGEMENT_MSG_INDICATION Primitive

Name :

S_MANAGEMENT_MSG_INDICATION

Arguments :

1. MSG

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.

A.2.1.17 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** (1) 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.

A.2.1.18 S_HARD_LINK_ESTABLISH Primitive

Name :

S_HARD_LINK_ESTABLISH

Arguments :

1. Link Priority
2. Link Type
3. Remote Node Address
4. Remote SAP ID

Direction :

Client-> Subnetwork Interface

Description :

The S_HARD_LINK_ESTABLISH primitive **shall** (1) be used by a client to request the establishment of a Hard Link between the local Node to which it is connected and a specified remote Node.

[Note: Physical Links between Nodes are normally made and broken unilaterally by the HF subnetwork according to the destinations of the queued U_PDUs. Such links are classified as Soft Links. The S_HARD_LINK_ESTABLISH primitive allows a client to override these procedures and request a Physical Link to be made to a specific Node and be maintained until the requesting client decides to break it.]

The argument *Link Priority* **shall** (2) define the priority of the Link. It **shall** (3) take a value in the range 0-3.

An S_HARD_LINK_ESTABLISH primitive with a higher Link Priority value **shall** (4) take precedence over a Hard Link established with a lower Link Priority value submitted by a client of the same Rank.

Hard Link requests made by clients with higher Rank **shall** (5) take precedence over requests of lower-Ranked clients regardless of the value of the *Link Priority* argument, in accordance with the requirements of Section A.3.2.2.1.

The *Link Type* argument **shall** (6) be used by the requesting client to fully or partially reserve the bandwidth of the Link. It **shall** (7) take a value in the range 0-2, as specified in Section A.1.1.2, specifying this primitive as one for a Type 0 Hard Link, Type 1 Hard Link, or Type 2 Hard Link, respectively.

The *Remote Node Address* argument **shall** (8) specify the physical HF Node Address to which the connection must be established and maintained.

The *Remote SAP ID* argument **shall** (9) identify the single client connected to the remote Node, to and from which traffic is allowed. This argument **shall** (10) be valid only if the *Link Type* argument has a value of 2 (i.e., only if the Hard Link request reserves the full bandwidth of the link for the local and remote client, as specified in section A.1.1.2.3).

A.2.1.19 S_HARD_LINK_TERMINATE Primitive

Name :

S_HARD_LINK_TERMINATE

Arguments :

1. Remote Node Address

Direction :

Client-> Subnetwork Interface

Description :

The S_HARD_LINK_TERMINATE primitive **shall** (1) be issued by a client to terminate an existing Hard Link.

The subnetwork **shall** (2) terminate an existing Hard Link on receipt of this primitive only if the primitive was generated by the client which requested the establishment of the Hard Link.

The single argument *Remote Node Address* **shall** (3) specify the Address of the Node at the remote end of the Hard Link.
[Note: The *Remote Node Address* argument is redundant in that Hard Links can exist with only one remote node at any time. It may however be used by the subnetwork implementation receiving the primitive to check its validity.]

Upon receiving this primitive, the subnetwork **shall** (4) take all necessary steps to terminate the Hard Link, as specified in section A.3.2.2.3[2].

[Note: The HARD LINK TERMINATE primitive is always accepted, and the subnetwork will terminate the link whether or not the remote node responds to the termination protocol. As specified in section A.3.2.2.3, the subnetwork will issue a S_HARD_LINK_TERMINATED primitive confirming the successful termination of the Link only if the termination protocol ends without confirmation from the remote note and the subnetwork was required to terminate the Hard Link unilaterally.]

A.2.1.20 S_HARD_LINK_ESTABLISHED Primitive

Name :
S_HARD_LINK_ESTABLISHED

Arguments :
1. Remote Node Status
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :
Subnetwork Interface-> Client

Description:
The S_HARD_LINK_ESTABLISHED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer as a positive response to a client's S_HARD_LINK_ESTABLISH primitive.

This primitive **shall** (2) be issued only after all the negotiations and protocols between the appropriate peer sublayers of the local and remote nodes have been completed and the remote node has accepted the establishment of the Hard Link, in accordance with the protocol specified in Section A.3.2.2.2.

The first argument, *Remote Node Status,* **shall** (3) inform the requesting client of any special status of the remote node, e.g. Remote Node in EMCON, etc. Valid arguments for *Remote Node Status* are given in the table below.

Remote Node Status	Value
ERROR	0
OK	>=1

Subsequent versions of this STANAG and implementation-dependent options may define additional values for the remote node status, for example, through use of the same set of local-node status codes defined for the S_SUBNET_AVAILABILITY primitive to report the status of a remote node. Successful establishment of a Hard Link **shall** (4) always imply a status of "OK" for the remote node; the value OK **shall** (5) be indicated by any positive non-zero value in the Remote Node Status field.

[2] The Link can be terminated immediately or in a "graceful" manner according to the requirements of a specific application and implementation. A graceful termination might, for example, allow completion of the current transmission interval before the link is broken and/or allow transmission of queued high priority U_PDUs from other clients to the same destination to be transmitted before the link is terminated.

The argument *Link Priority* **shall** (6) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

The *Link Type* argument **shall** (7) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

The *Remote Node Address* argument **shall** (8) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

The *Remote SAP ID* argument **shall** (9) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

A.2.1.21 S_HARD_LINK_REJECTED Primitive

Name :

S_HARD_LINK_REJECTED

Arguments :

1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

The S_HARD_LINK_REJECTED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer as a negative response to a client's S_HARD_LINK_ESTABLISH primitive.

The *Reason* argument **shall** (2) specify why the Hard Link Request was rejected, with values defined for this argument as specified in the table below:

Reason	Value
Remote-Node-Busy	1
Higher-Priority-Link-Existing	2
Remote-Node-Not-Responding	3
Destination SAP ID not bound	4
Requested Type 0 Link Exists	5

The argument *Link Priority* **shall** (3) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ REJECTED Primitive is the response.

The *Link Type* argument **shall** (4) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ REJECTED Primitive is the response.

The *Remote Node Address* argument **shall** (5) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ REJECTED Primitive is the response.

The *Remote SAP ID* argument **shall** (6) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ REJECTED Primitive is the response.

A.2.1.22 S_HARD_LINK_TERMINATED Primitive

Name :

S_HARD_LINK_TERMINATED

Arguments :

1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

The S_HARD_LINK_TERMINATED primitive **shall** (1) be issued by the Subnetwork Interface Sublayer to inform a client which has been granted a Hard Link that the Link has been terminated unilaterally by the Subnetwork.

For Hard Link Types 0 and 1, only the client that originally requested the Hard Link **shall**(2) receive this primitive. Other clients sharing the link with Soft-Link Data Exchange Sessions may have the link broken without notification.

For type 2 hard links, both called and calling clients **shall** (3) receive this primitive.

The *Reason* argument **shall** (4) specify why the Hard Link was terminated, with values defined for this argument as specified in the table below:

Reason	Value
Link terminated by remote node	1
Higher priority link requested	2
Remote node not responding (time out)	3
Destination SAP ID unbound	4
Physical Link Broken	5

The argument *Link Priority* **shall** (5) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ TERMINATED Primitive is the response.

The *Link Type* argument **shall** (6) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ TERMINATED Primitive is the response.

The *Remote Node Address* argument **shall** (7) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ TERMINATED Primitive is the response.

The *Remote SAP ID* argument **shall** (8) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ TERMINATED Primitive is the response.

A.2.1.23 S_HARD_LINK_INDICATION Primitive

Name :

S_HARD_LINK_INDICATION

Arguments :

1. Remote Node Status
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

The S_HARD_LINK_INDICATION primitive **shall** (1) be used only for Hard Link Type 2. With this primitive the Subnetwork Interface Sublayer **shall** (2) signal to one of its local clients that a client at a remote node requested a Hard Link of Type 2 to be established between them.

The first argument, *Remote Node Status*, **shall** (3) inform the local client of any special status of the remote node, e.g. Remote Node in EMCON, etc. Valid arguments currently defined for *Remote Node Status* are given in the table below.

Remote Node Status	Value
ERROR	0
OK	>=1

Subsequent versions of this STANAG and implementation-dependent options may define additional values for the remote node status. At present, a minimally compliant client implementation may ignore this argument.

The argument *Link Priority* **shall** (4) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive generated by the remote-client and for which this S_HARD_LINK_ INDICATION Primitive is the result.

The *Link Type* argument **shall** (5) have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive generated by the remote-client and for which this S_HARD_LINK_ INDICATION Primitive is the result.

The *Remote Node Address* argument **shall** (6) be equal in value to the HF subnetwork address of the node to which the remote-client is bound and that originated the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_INDICATION Primitive is the result.

The *Remote SAP ID* argument **shall** (7) be equal in value to the SAP_ID that is bound to the remote client that originated the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ INDICATION Primitive is the result.

A.2.1.24 S_HARD_LINK_ACCEPT Primitive

Name :

S_HARD_LINK_ACCEPT

Arguments :

1. Link Priority
2. Link Type
3. Remote Node Address
4. Remote SAP ID

Direction :

Client-> Subnetwork Interface

Description:

The S_HARD_LINK_ACCEPT primitive **shall** (1) be issued by a client as a positive response to a S_HARD_LINK_INDICATION primitive. With this primitive the client tells the Subnetwork Interface Sublayer that it accepts the Hard Link of Type 2 requested by a client at a remote node.

The argument *Link Priority* **shall** (2) have the same meaning and be equal in value to the *Link Priority* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ ACCEPT Primitive is the response.

The *Link Type* argument **shall** (3) have the same meaning and be equal in value to the *Link Type* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ ACCEPT Primitive is the response.

The *Remote Node Address* argument **shall** (4) have the same meaning and be equal in value to the *Remote Node Address* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ ACCEPT Primitive is the response.

The *Remote SAP ID* argument **shall** (5) have the same meaning and be equal in value to the *Remote SAP ID* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ ACCEPT Primitive is the response.

A.2.1.25 S_HARD_LINK_REJECT Primitive

Name :

S_HARD_LINK_REJECT

Arguments :

1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Client-> Subnetwork Interface

Description:

The S_HARD_LINK_REJECT primitive **shall** (1) be issued by a client as a negative response to a S_HARD_LINK_INDICATION primitive. With this primitive the client tells the Subnetwork Interface Sublayer that it rejects the Hard Link of Type 2 requested by a client at a remote node.

The *Reason* argument **shall** (2) specify why the hard link is rejected. Possible values of this argument are Mode-Not-Supported (for Link Type 2), I-Have-Higher-Priority-Data, etc.

The argument *Link Priority* **shall** (3) have the same meaning and be equal in value to the *Link Priority* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_REJECT Primitive is the response.

The *Link Type* argument **shall** (4) have the same meaning and be equal in value to the *Link Type* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_REJECT Primitive is the response.

The *Remote Node Address* argument **shall** (5) have the same meaning and be equal in value to the *Remote Node Address* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_REJECT Primitive is the response.

The *Remote SAP ID* argument **shall** (6) have the same meaning and be equal in value to the *Remote SAP ID* argument of the S_HARD_LINK_ INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ ACCEPT Primitive is the response.

A.2.1.26 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 by a high ranked client. 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. If the Subnetwork Status is other than ON, the *Reason* argument explains why. Possible values of this argument are Local-Node-In-EMCON, Hard-Link-Requested-By High-Priority-client, etc.

A.2.2 Encoding of Primitives

The encoding of the S_Primitives for communication across the Subnetwork Interface Sublayer **shall** (1) be in accordance with text and figures in the subsections below.

A.2.2.1 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 defined in C.3.1.4. 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 Annex C for D_PDUs.

A.2.2.2 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.

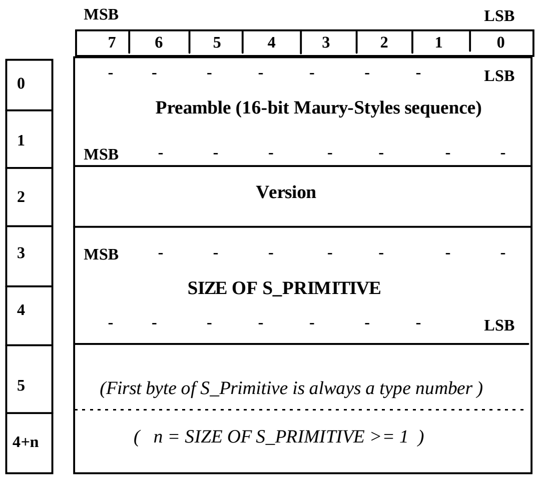

MSB
LSB
7 6 5 4 3 2 1 0
0
1
2
3
4
5
4+n
LSB
Preamble (16-bit Maury-Styles sequence)
MSB
Version
MSB
SIZE OF S_PRIMITIVE
LSB
(First byte of S_Primitive is always a type number)
(n = SIZE OF S_PRIMITIVE >= 1)

Figure A-1(a): 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:

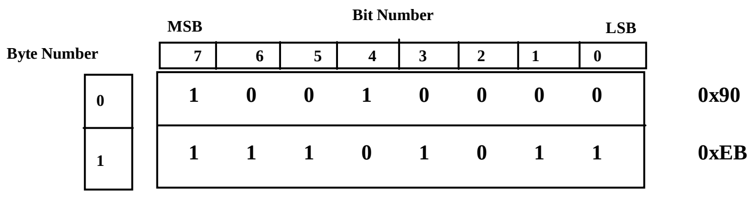


Figure A-1(b): Encoding of Maury-Styles Preamble-Sequence in "S_" Primitives

[***Note***: This encoding of the Maury-Styles preamble sequence is an exception to the general requirement of section 2.2.1 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 follows:

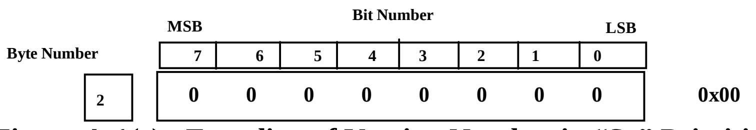


Figure A-1(c): Encoding of Version Number in "S_" Primitives

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** (8) be mapped into the low order bit of the low-order byte of the field, as follows:

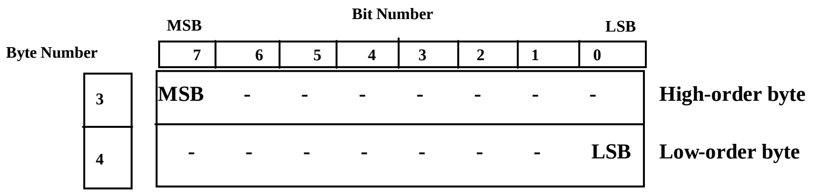


Figure A-1(c): Encoding of Size_of_S_Primitive Element in "S_" Primitives

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** (12) 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.

A.2.2.3 S_BIND_REQUEST Encoding

The S_BIND_REQUEST primitive **shall** (1) be encoded as a four-byte field as follows:

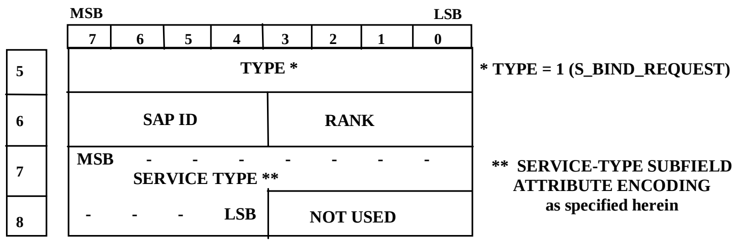
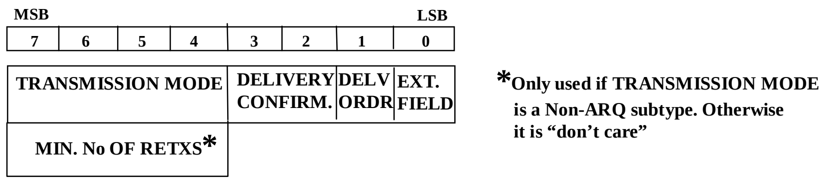


Figure A-2: Encoding of S_BIND_REQUEST Primitive

The S_BIND_REQUEST SERVICE-TYPE field **shall** (2) be encoded as five subfields as follows:

S_BIND_REQUEST: SUBFIELD ENCODING OF “SE RVICE TYPE” FIELD



TRANSMISSION MODE	DELIVERY CONFIRMATION	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore Service type field.	00 = No Confirmation	0 = In-order Delivery	0 = No extended field
0001 = ARQ	01 = Node Delivery Conf.	1 = As they arrive	1 = Extended field follows
0010 = Non-ARQ (Broadcast)	10 = Client Delievery Conf.		
0011-.... = other Non-ARQ types etc.	11 = Not defined		

Figure A-3: 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 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.]

A.2.2.4 S_UNBIND_REQUEST Encoding

The S_UNBIND_REQUEST primitive **shall** (1) be encoded as a one-byte field as follows:

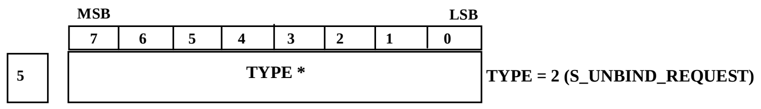


Figure A-4: Encoding of S_UNBIND_REQUEST Primitive

A.2.2.5 S_BIND_ACCEPTED Encoding

The S_BIND_ACCEPTED primitive **shall** (1) be encoded as a four-byte field as follows:

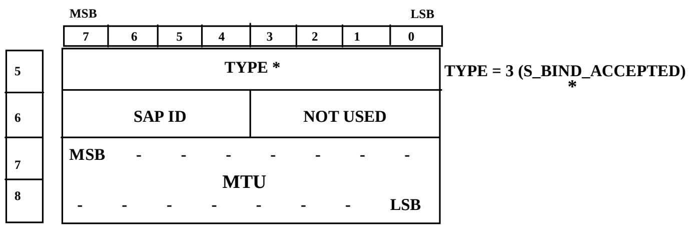


Figure A-5: Encoding of S_BIND_ACCEPTED Primitive

A.2.2.6 S_BIND_REJECTED Encoding

The S_BIND_REJECTED primitive **shall** (1) be encoded as a two-byte field as follows:

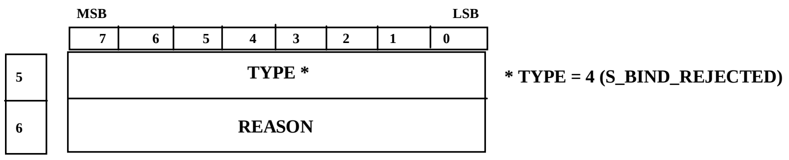


Figure A-6: Encoding of S_BIND_REJECTED Primitive

A.2.2.7 S_UNBIND_INDICATION Encoding

The S_UNBIND_INDICATION primitive **shall** (1) be encoded as a two-byte field as follows:

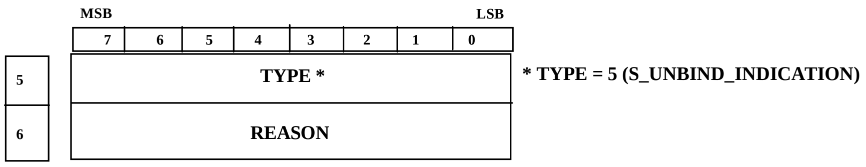


Figure A-7: Encoding of S_UNBIND_INDICATION Primitives

A.2.2.8 S_HARD_LINK_ESTABLISH Encoding

The S_HARD_LINK_ESTABLISH primitive **shall** (1) be encoded as a six-byte field as follows:

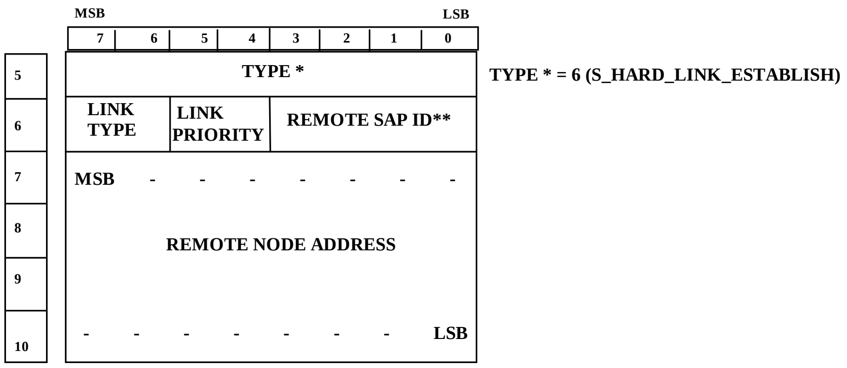


Figure A-8: Encoding of S_HARD_LINK_ESTABLISH Primitives

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.9 S_HARD_LINK_TERMINATE Encoding

The S_HARD_LINK_TERMINATE primitive **shall** (1) be encoded as a five-byte field as follows:

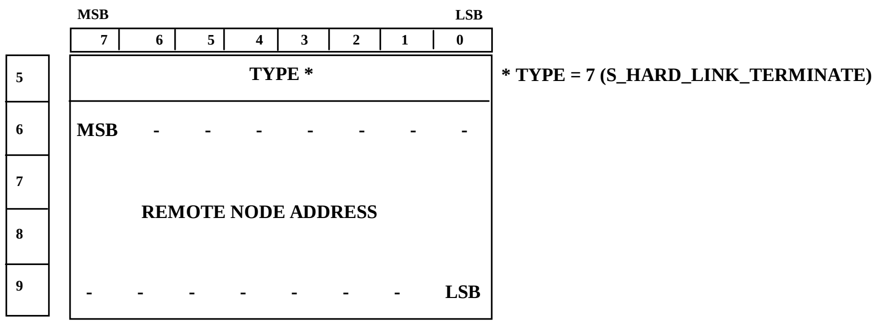


Figure A-9: Encoding of S_HARD_LINK_TERMINATE Primitives

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.10 S_HARD_LINK_ESTABLISHED Encoding

The S_HARD_LINK_ESTABLISHED primitive **shall** (1) be encoded as a seven-byte field as follows:

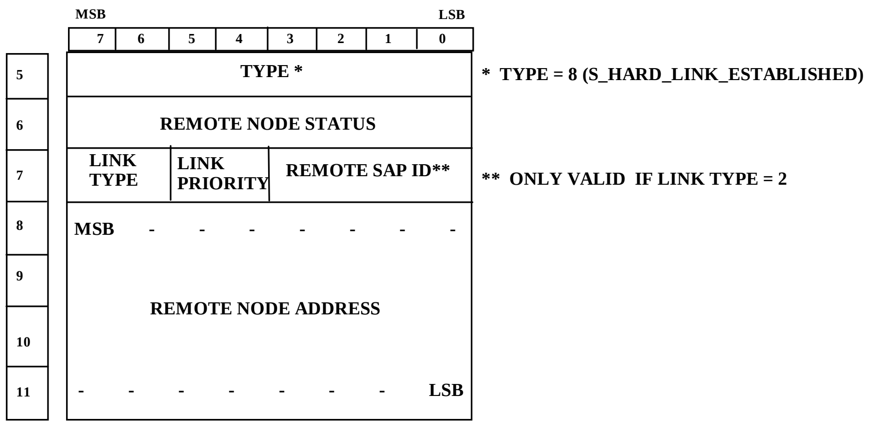


Figure A-10: Encoding of S_HARD_LINK_ESTABLISHED Primitives.

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.11 S_HARD_LINK_REJECTED Encoding

The S_HARD_LINK_REJECTED primitive **shall** (1) be encoded as a seven-byte field as follows:

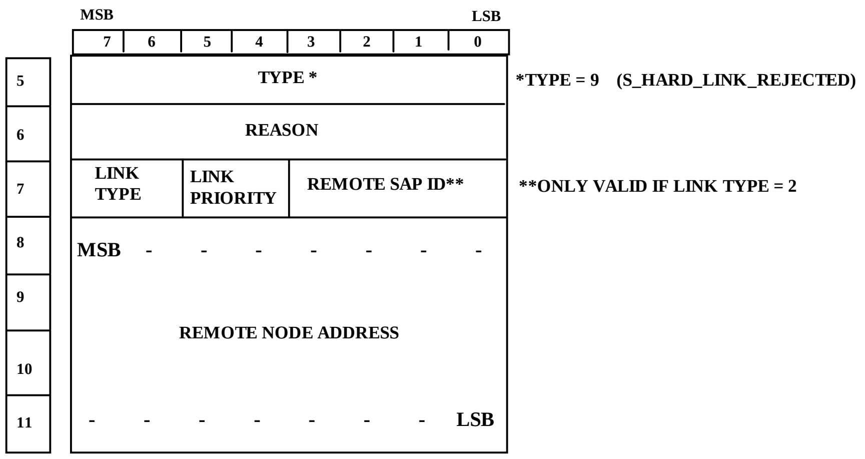


Figure A-11: Encoding of S_HARD_LINK_REJECTED Primitives.

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.12 S_HARD_LINK_TERMINATED Encoding

The S_HARD_LINK_TERMINATED primitive **shall** (1) be encoded as a seven-byte field as follows:

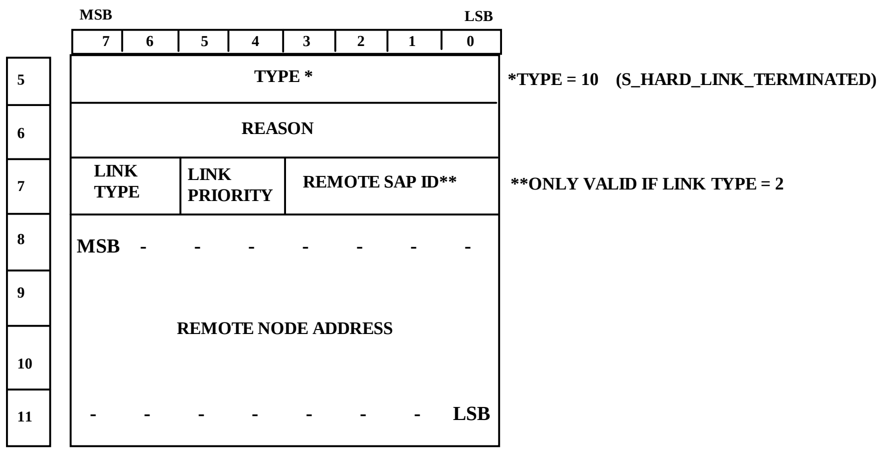


Figure A-12: Encoding of S_HARD_LINK_TERMINATED Primitives.

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.13 S_HARD_LINK_INDICATION Encoding

The S_HARD_LINK_INDICATION primitive **shall** (1) be encoded as a seven-byte field as follows:

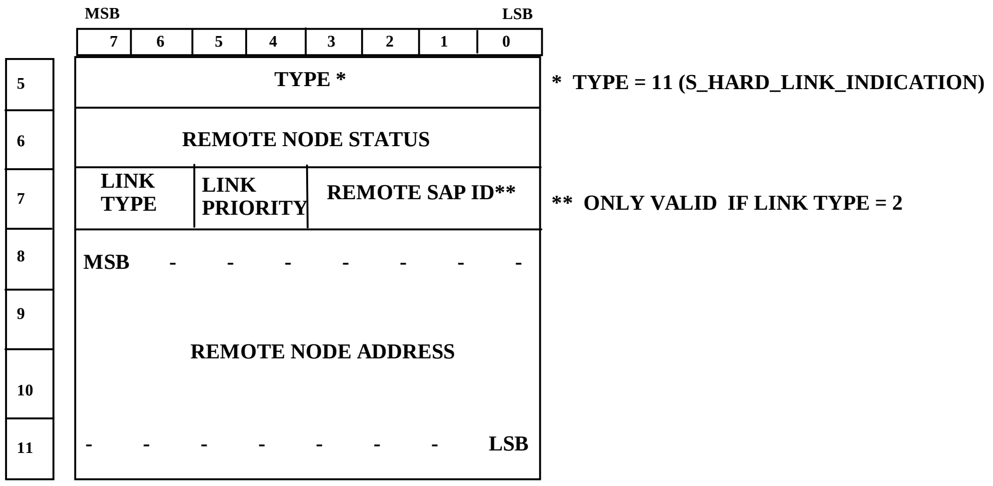


Figure A-13: Encoding of S_HARD_LINK_INDICATION Primitives.

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.14 S_HARD_LINK_ACCEPT Encoding

The S_HARD_LINK_ACCEPT primitive **shall** (1) be encoded as a six-byte field as follows:

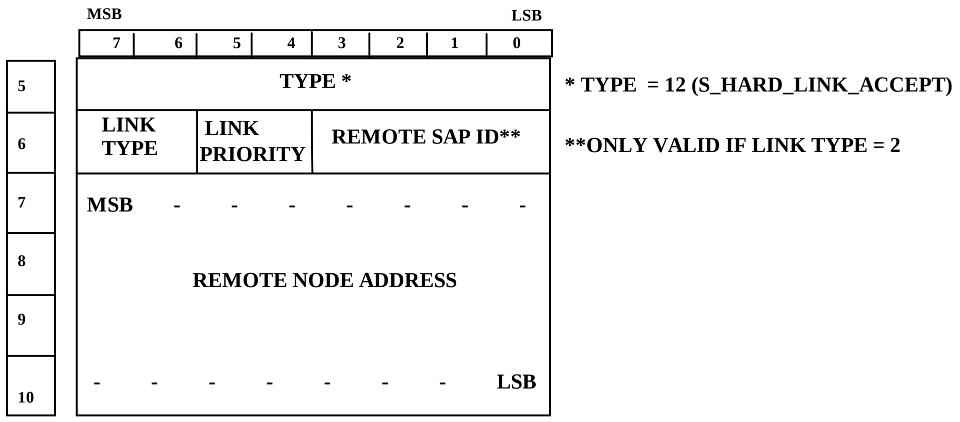


Figure A-14: Encoding of S_HARD_LINK_ACCEPT Primitives

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.15 S_HARD_LINK_REJECT Encoding

The S_HARD_LINK_REJECT primitive **shall** (1) be encoded as a seven-byte field as follows:

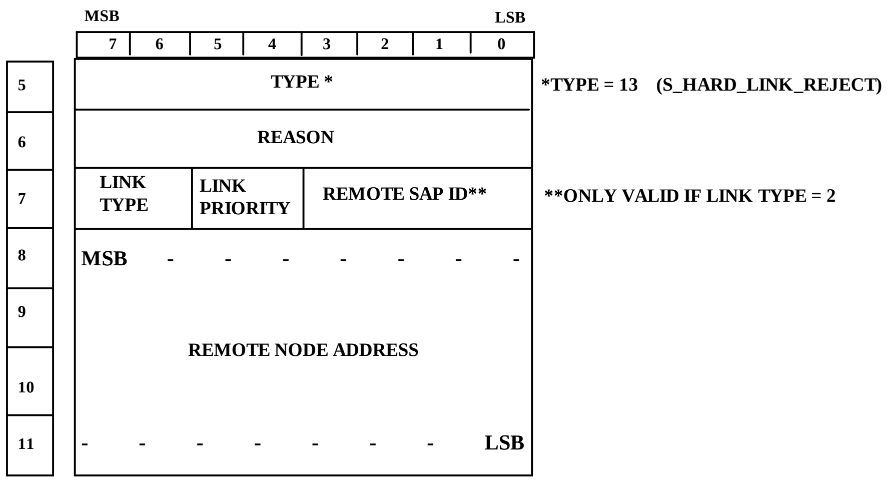


Figure A-15: Encoding of S_HARD_LINK_REJECT Primitives.

The REMOTE NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** (3) be encoded as specified in Section A.2.2.28.4.

A.2.2.16 S_SUBNET_AVAILABILITY Encoding

The S_SUBNET_AVAILABILITY primitive **shall** (1) be encoded as a three-byte field as follows:

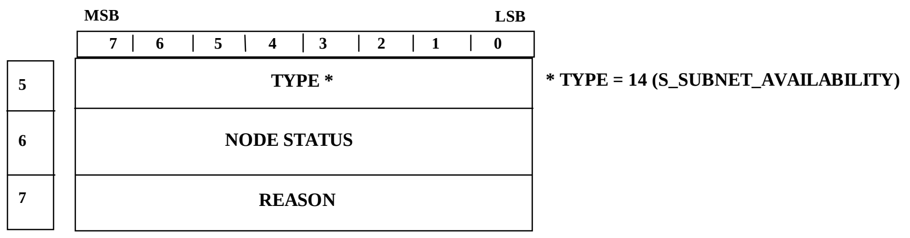


Figure A-16: Encoding of S_SUBNET_AVAILABILITY Primitives.

The encoding of the NODE STATUS and REASON fields is implementation dependent.

A.2.2.17 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:

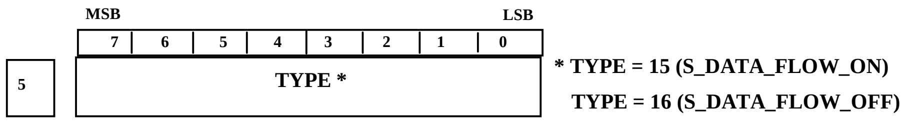


Figure A-17: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

A.2.2.18 S_KEEP_ALIVE Encoding

The S_KEEP_ALIVE primitive **shall** (1) be encoded as a one-byte field as follows:

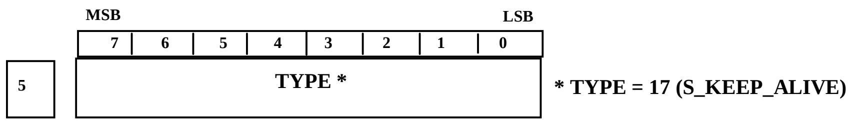


Figure A-18: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

A.2.2.19 S_MANAGEMENT_MESSAGE_REQUEST and S_MANAGEMENT_MESSAGE_INDICATION Encoding

The S_MANAGEMENT_MESSAGE_REQUEST and S_MANAGEMENT_MESSAGE_INDICATION primitives **shall** (1) be encoded as implementation-dependent variable-length fields as follows:

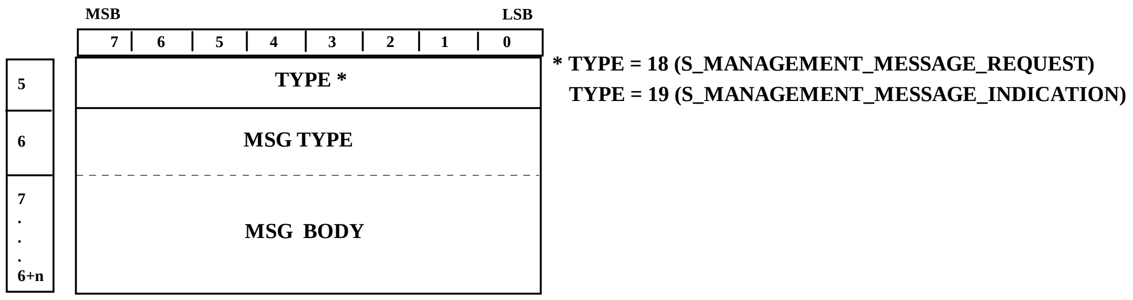

MSB
LSB
7
6
5
4
3
2
1
0
5
TYPE *
6
MSG TYPE
7 . . . 6+n
MSG BODY
* TYPE = 18 (S_MANAGEMENT_MESSAGE_REQUEST)
TYPE = 19 (S_MANAGEMENT_MESSAGE_INDICATION)

Figure A-19: Encoding of S_MANAGEMENT_MESSAGE_REQUEST and S_MANAGEMENT_MESSAGE_INDICATION Primitives.

The encoding of the MSG TYPE and MSG BODY fields is implementation dependent.

A.2.2.20 S_UNIDATA_REQUEST Encoding

The S_UNIDATA_REQUEST primitive **shall** (1) be encoded as a variable-length field as follows:

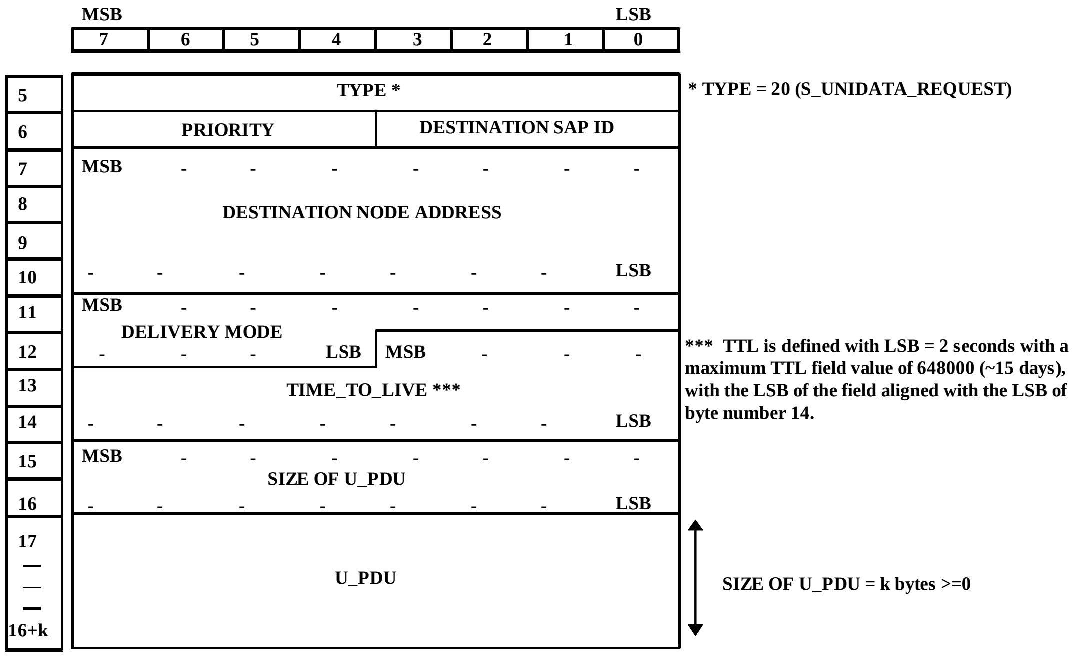

MSB
LSB
7
6
5
4
3
2
1
0
5
6
7
8
9
10
11
12
13
14
15
16
17
16+k
TYPE *
* TYPE = 20 (S_UNIDATA_REQUEST)
PRIORITY
DESTINATION SAP ID
DESTINATION NODE ADDRESS
DELIVERY MODE
TIME_TO_LIVE ***
*** TTL is defined with LSB = 2 seconds with a maximum TTL field value of 648000 (~15 days), with the LSB of the field aligned with the LSB of byte number 14.
SIZE OF U_PDU
U_PDU
SIZE OF U_PDU = k bytes >=0

Figure A-20: Encoding of S_UNIDATA_REQUEST Primitives.

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in Section A.2.2.28.1.

The DELIVERY MODE field **shall** (3) be encoded as specified in Section A.2.2.28.2.

A.2.2.21 S_UNIDATA_INDICATION Encoding

The S_UNIDATA_INDICATION primitive **shall** (1) be encoded as a variable-length field as follows:

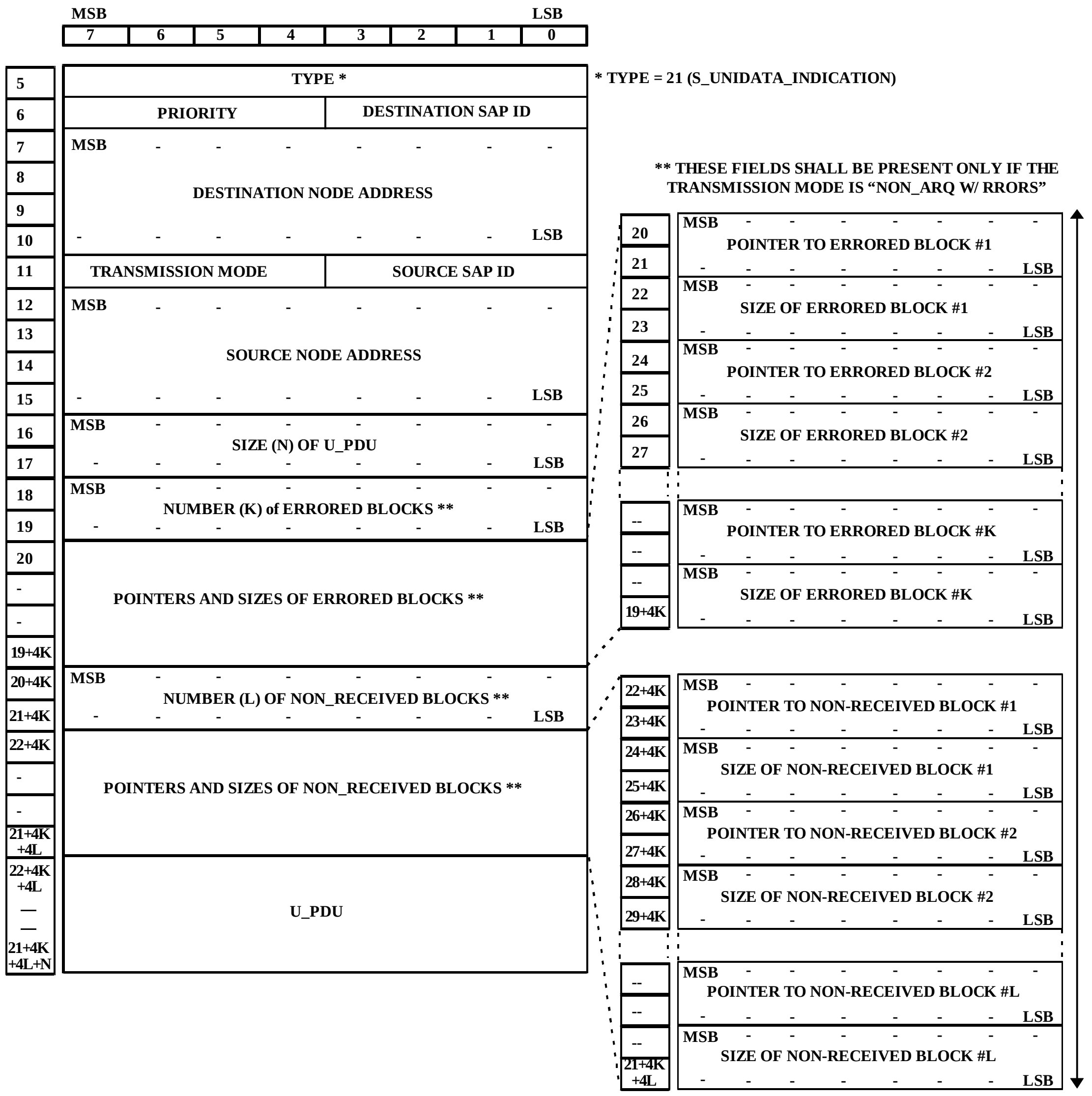

MSB
LSB
7 6 5 4 3 2 1 0
TYPE *
* TYPE = 21 (S_UNIDATA_INDICATION)
PRIORITY
DESTINATION SAP ID
DESTINATION NODE ADDRESS
TRANSMISSION MODE
SOURCE SAP ID
SOURCE NODE ADDRESS
SIZE (N) OF U_PDU
NUMBER (K) of ERRORED BLOCKS **
POINTERS AND SIZES OF ERRORED BLOCKS **
NUMBER (L) OF NON_RECEIVED BLOCKS **
POINTERS AND SIZES OF NON_RECEIVED BLOCKS **
U_PDU
** THESE FIELDS SHALL BE PRESENT ONLY IF THE TRANSMISSION MODE IS “NON_ARQ W/ RRORS”
POINTER TO ERRORED BLOCK #1
SIZE OF ERRORED BLOCK #1
POINTER TO ERRORED BLOCK #2
SIZE OF ERRORED BLOCK #2
POINTER TO ERRORED BLOCK #K
SIZE OF ERRORED BLOCK #K
POINTER TO NON-RECEIVED BLOCK #1
SIZE OF NON-RECEIVED BLOCK #1
POINTER TO NON-RECEIVED BLOCK #2
SIZE OF NON-RECEIVED BLOCK #2
POINTER TO NON-RECEIVED BLOCK #L
SIZE OF NON-RECEIVED BLOCK #L
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
19+4K 20+4K 21+4K 22+4K 21+4K +4L 22+4K +4L 21+4K +4L+N
20 21 22 23 24 25 26 27
22+4K 23+4K 24+4K 25+4K 26+4K 27+4K 28+4K 29+4K

Figure A-21: Encoding of S_UNIDATA_INDICATION Primitives

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in Section A.2.2.28.1.

The TRANSMISSION MODE field **shall** (3) be encoded as specified in Section A.2.2.28.3.

A.2.2.22 S_UNIDATA_CONFIRM Encoding

The S_UNIDATA_CONFIRM primitive **shall** (1) be encoded as a variable-length field as follows:

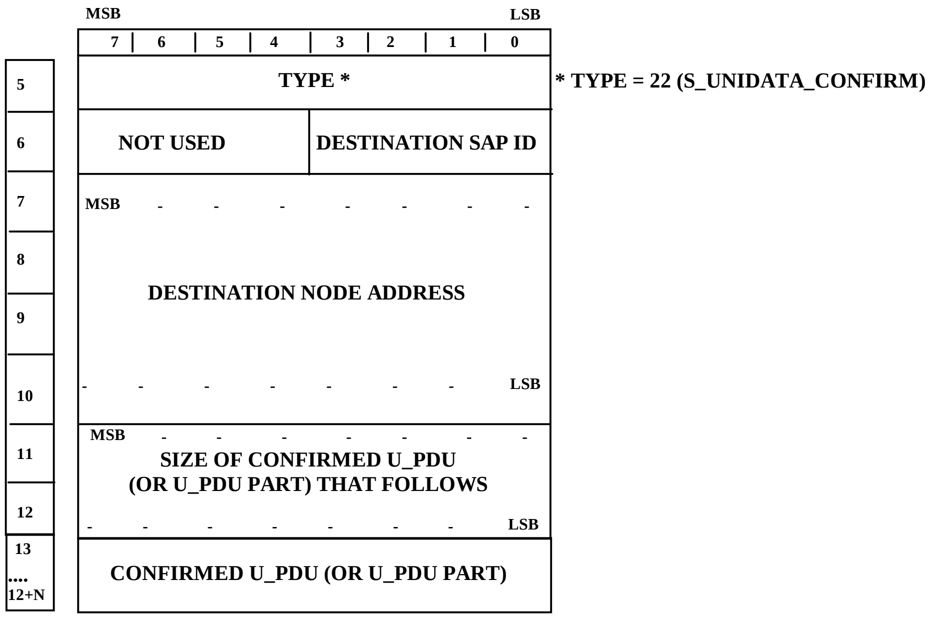

MSB
LSB
7
6
5
4
3
2
1
0
5
6
7
8
9
10
11
12
13
....
12+N
TYPE *
* TYPE = 22 (S_UNIDATA_CONFIRM)
NOT USED
DESTINATION SAP ID
MSB
DESTINATION NODE ADDRESS
LSB
MSB
SIZE OF CONFIRMED U_PDU (OR U_PDU PART) THAT FOLLOWS
LSB
CONFIRMED U_PDU (OR U_PDU PART)

Figure A-22: Encoding of S_UNIDATA_CONFIRM Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

A.2.2.23 S_UNIDATA_REJECTED Encoding

The S_UNIDATA_REJECTED primitive **shall** (1) be encoded as a variable-length field as follows:

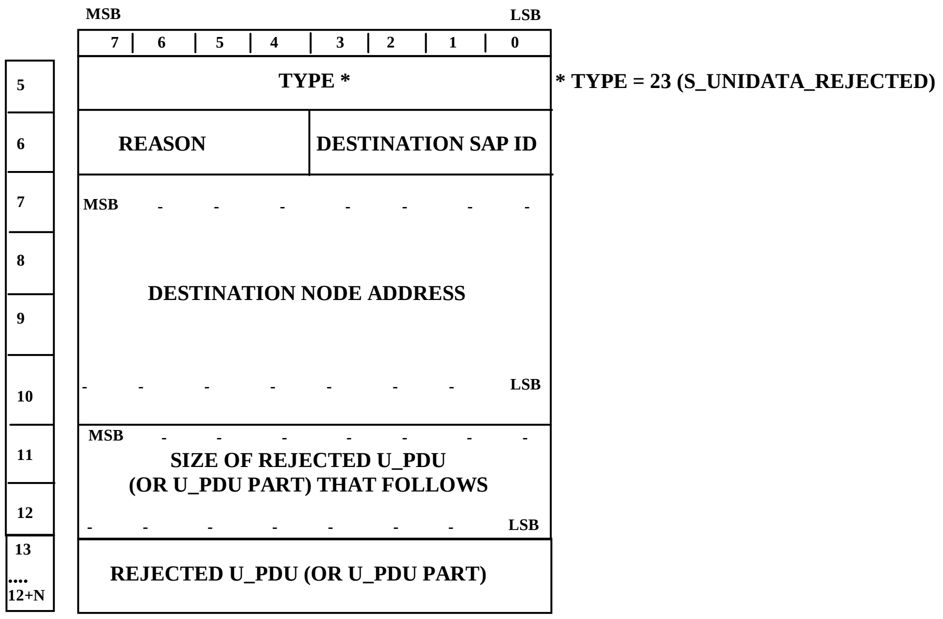

MSB
LSB
7 6 5 4 3 2 1 0
5
6
7
8
9
10
11
12
13
....
12+N
TYPE *
* TYPE = 23 (S_UNIDATA_REJECTED)
REASON
DESTINATION SAP ID
MSB
DESTINATION NODE ADDRESS
LSB
MSB
SIZE OF REJECTED U_PDU (OR U_PDU PART) THAT FOLLOWS
LSB
REJECTED U_PDU (OR U_PDU PART)

Figure A-23: Encoding of S_UNIDATA_REJECTED Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

A.2.2.24 S_EXPEDITED_UNIDATA_REQUEST Encoding

The S_EXPEDITED_UNIDATA_REQUEST primitive **shall** (1) be encoded as a variable-length field as follows:

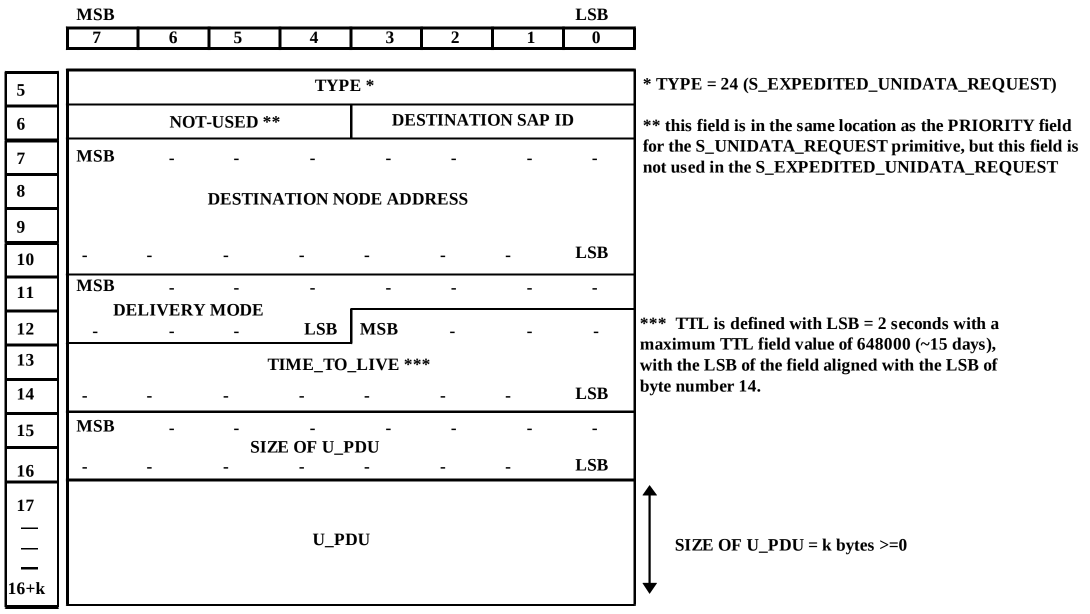

MSB
LSB
7
6
5
4
3
2
1
0
5
6
7
8
9
10
11
12
13
14
15
16
17
16+k
TYPE *
NOT-USED **
DESTINATION SAP ID
MSB
DESTINATION NODE ADDRESS
LSB
MSB
DELIVERY MODE
LSB
MSB
TIME_TO_LIVE ***
LSB
MSB
SIZE OF U_PDU
LSB
U_PDU
* TYPE = 24 (S_EXPEDITED_UNIDATA_REQUEST)
** this field is in the same location as the PRIORITY field for the S_UNIDATA_REQUEST primitive, but this field is not used in the S_EXPEDITED_UNIDATA_REQUEST
*** TTL is defined with LSB = 2 seconds with a maximum TTL field value of 648000 (~15 days), with the LSB of the field aligned with the LSB of byte number 14.
SIZE OF U_PDU = k bytes >=0

Figure A-24: Encoding of S_EXPEDITED_UNIDATA_REQUEST Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

The DELIVERY MODE field **shall** (3) be encoded as specified in Section A.2.2.28.2.

A.2.2.25 S_EXPEDITED_UNIDATA_INDICATION Encoding

The S_EXPEDITED_UNIDATA_INDICATION primitive **shall** (1) be encoded as a variable-length field as follows:

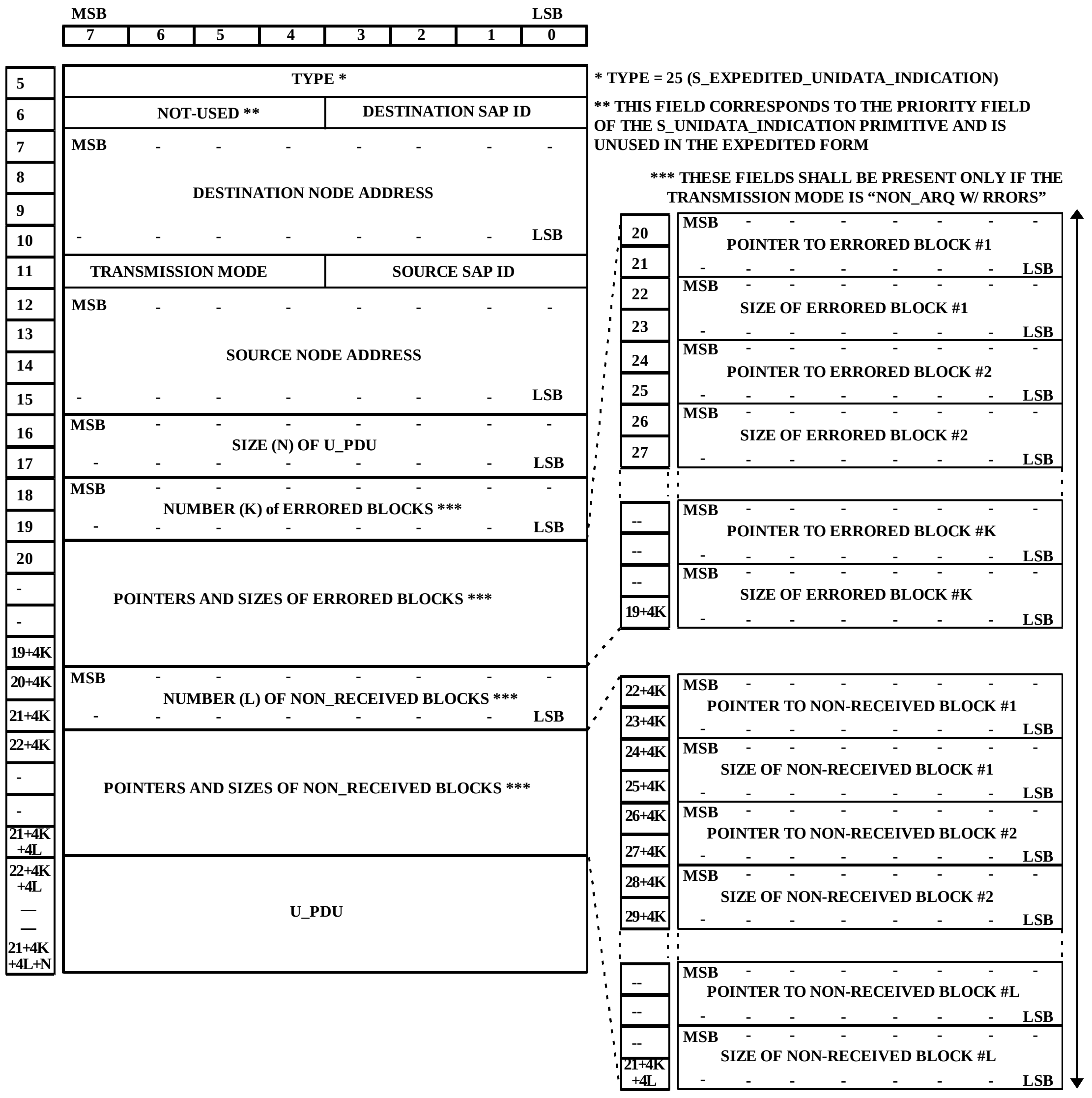

MSB
LSB
7 6 5 4 3 2 1 0
TYPE *
NOT-USED **
DESTINATION SAP ID
DESTINATION NODE ADDRESS
TRANSMISSION MODE
SOURCE SAP ID
SOURCE NODE ADDRESS
SIZE (N) OF U_PDU
NUMBER (K) of ERRORED BLOCKS ***
POINTERS AND SIZES OF ERRORED BLOCKS ***
NUMBER (L) OF NON_RECEIVED BLOCKS ***
POINTERS AND SIZES OF NON_RECEIVED BLOCKS ***
U_PDU
5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 19+4K 20+4K 21+4K 22+4K 21+4K +4L 22+4K +4L 21+4K +4L+N
* TYPE = 25 (S_EXPEDITED_UNIDATA_INDICATION)
** THIS FIELD CORRESPONDS TO THE PRIORITY FIELD OF THE S_UNIDATA_INDICATION PRIMITIVE AND IS UNUSED IN THE EXPEDITED FORM
*** THESE FIELDS SHALL BE PRESENT ONLY IF THE TRANSMISSION MODE IS "NON_ARQ W/ RRORS"
POINTER TO ERRORED BLOCK #1
SIZE OF ERRORED BLOCK #1
POINTER TO ERRORED BLOCK #2
SIZE OF ERRORED BLOCK #2
POINTER TO ERRORED BLOCK #K
SIZE OF ERRORED BLOCK #K
POINTER TO NON-RECEIVED BLOCK #1
SIZE OF NON-RECEIVED BLOCK #1
POINTER TO NON-RECEIVED BLOCK #2
SIZE OF NON-RECEIVED BLOCK #2
POINTER TO NON-RECEIVED BLOCK #L
SIZE OF NON-RECEIVED BLOCK #L
20 21 22 23 24 25 26 27 19+4K
22+4K 23+4K 24+4K 25+4K 26+4K 27+4K 28+4K 29+4K 21+4K +4L

Figure A-25: Encoding of S_EXPEDITED_UNIDATA_INDICATION Primitives

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in Section A.2.2.28.1.

The TRANSMISSION MODE field **shall** (3) be encoded as specified in Section A.2.2.28.3.

A.2.2.26 S_EXPEDITED_UNIDATA_CONFIRM Encoding

The S_EXPEDITED_UNIDATA_CONFIRM primitive **shall** (1) be encoded as a variable-length field as follows:

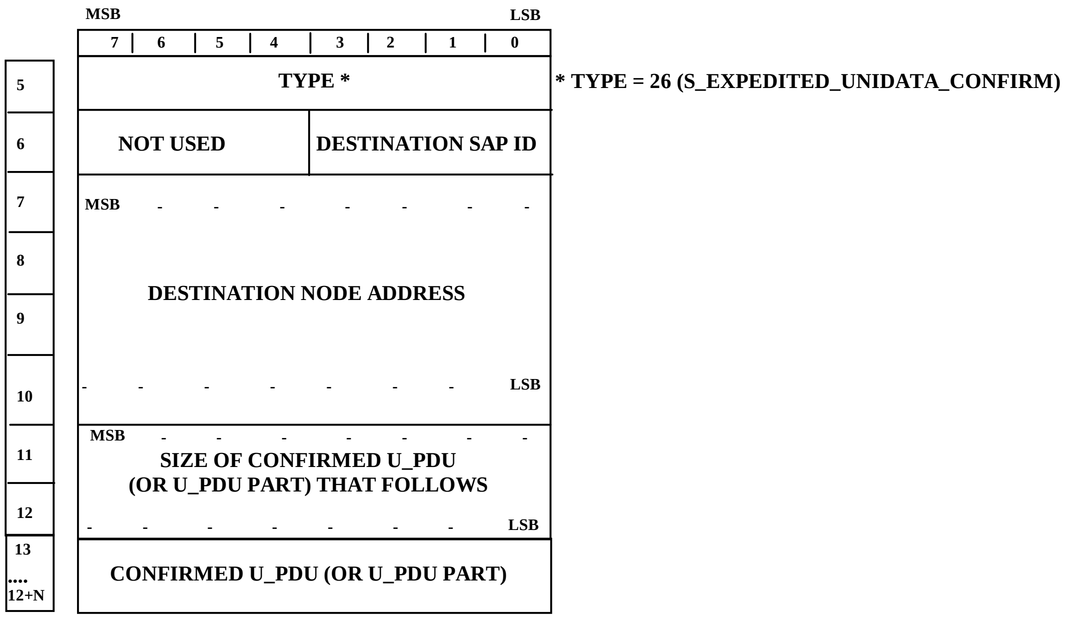


Figure A-26: Encoding of S_EXPEDITED_UNIDATA_CONFIRM Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

A.2.2.27 S_EXPEDITED_UNIDATA_REJECTED Encoding

The S_EXPEDITED_UNIDATA_REJECTED primitive **shall** (1) be encoded as a variable-length field as follows:

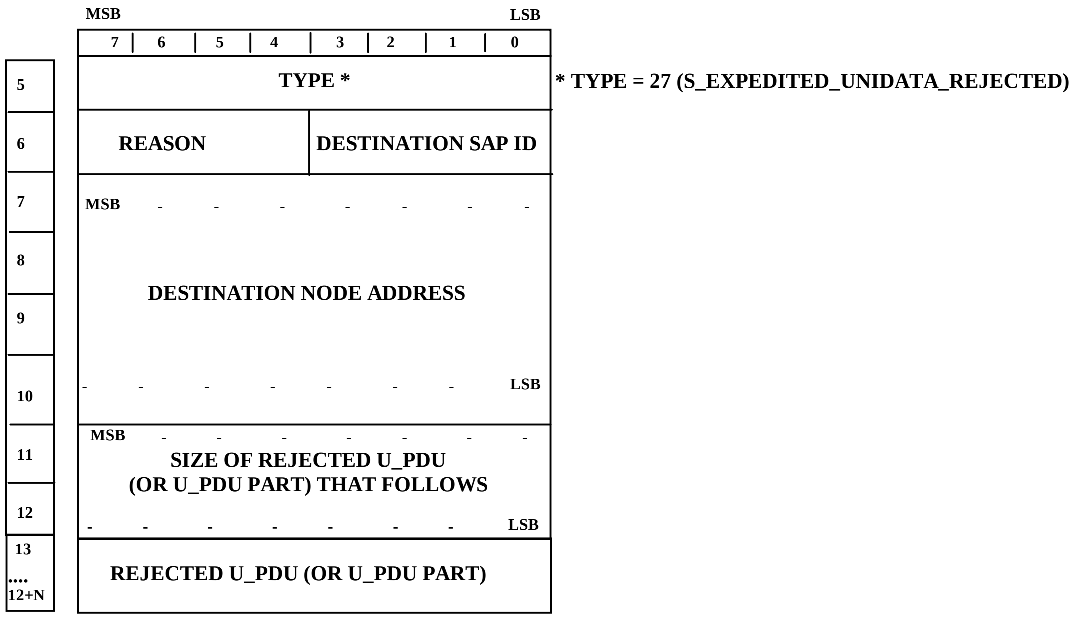


Figure A-27: Encoding of S_EXPEDITED_UNIDATA_REJECTED Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section A.2.2.28.1.

A.2.2.28 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.

A.2.2.28.1 Node ADDRESS Encoding for the all Primitives

Arguments : SOURCE NODE ADDRESS, DESTINATION NODE ADDRESS, REMOTE NODE ADDRESS
Primitives : ALL “UNIDATA” primitives and “S_HARD_LINK” 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 A-28.

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.

Remote node addresses and remote SAP IDs of all "S_HARD_LINK" primitives **shall** (3) be individual SAP IDs and individual node addresses respectively.

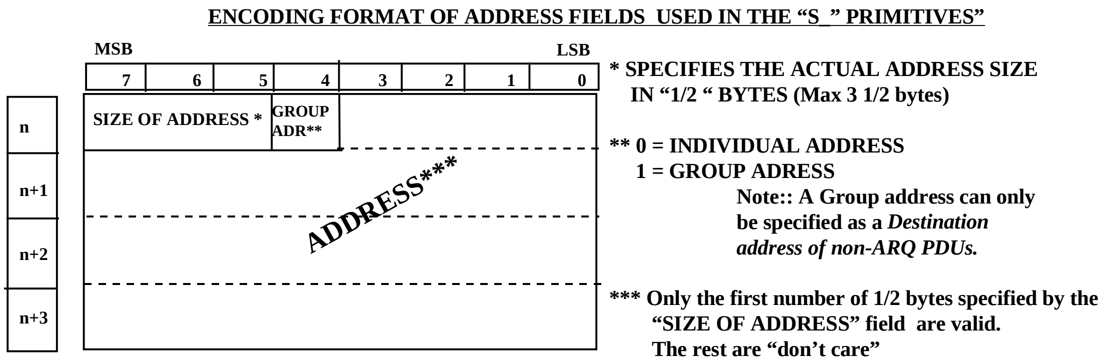

ENCODING FORMAT OF ADDRESS FIELDS USED IN THE "S_" PRIMITIVES"
MSB
LSB
7
6
5
4
3
2
1
0
n
n+1
n+2
n+3
SIZE OF ADDRESS *
GROUP ADR**
ADDRESS***
* SPECIFIES THE ACTUAL ADDRESS SIZE IN "1/2 " BYTES (Max 3 1/2 bytes)
** 0 = INDIVIDUAL ADDRESS
1 = GROUP ADRESS
Note:: A Group address can only be specified as a *Destination address of non-ARQ PDUs.*
*** Only the first number of 1/2 bytes specified by the "SIZE OF ADDRESS" field are valid. The rest are "don't care"

Figure A-28: Encoding of Address Fields in S_Primitives.

A.2.2.28.2 Delivery-Mode Encoding for the S_UNIDATA_REQUEST and S_EXPEDITED_UNIDATA_REQUEST Primitives

Argument : DELIVERY MODE
Primitive : S_UNIDATA_REQUEST , S_EXPEDITED_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 A-29.

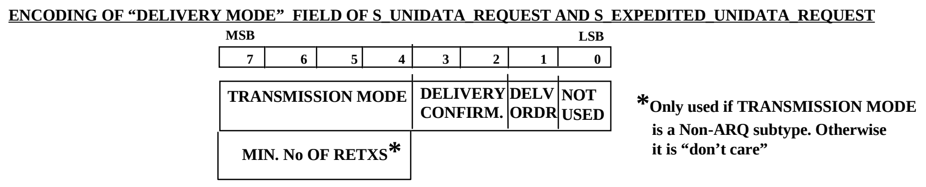
The value of the DELIVERY MODE argument can be "DEFAULT", as encoded by the Transmission Mode attribute. With a value of "DEFAULT", the delivery mode for this U_PDU **shall** (2) be the delivery mode specified in the *Service Type* argument of the S_BIND_REQUEST. A non-DEFAULT value **shall** (3) override 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:

6. *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.
7. *Data Delivery Confirmation for this PDU* --- None, node-to-node, or client-to-client.
8. *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.

9. *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".
10. *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.]

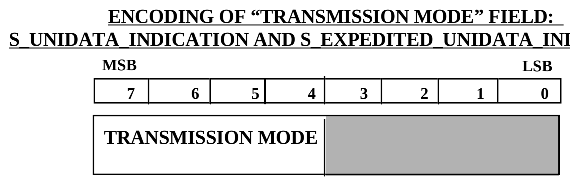

ENCODING OF "DELIVERY MODE" FIELD OF S_UNIDATA_REQUEST AND S_EXPEDITED_UNIDATA_REQUEST
MSB
LSB
7
6
5
4
3
2
1
0
TRANSMISSION MODE
DELIVERY CONFIRM.
DELV ORDR
NOT USED
MIN. No OF RETXS*
*Only used if TRANSMISSION MODE is a Non-ARQ subtype. Otherwise it is "don't care"

TRANSMISSION MODE	DELIVERY CONFIRM.	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore this field. Use default mode as specified during *S_BIND_REQUEST* time	00 = No Confirmation	0 = In-order Delivery	0 = No extended field
0001 = ARQ	01 = Node Delivery Conf.	1 = As they arrive	1 = Extended field follows
0010 = Non-ARQ (Broadcast)	10 = Client Delievery Conf.		
0011-.... = other Non-ARQ types, etc.	11 = Not defined		

Figure A-29: Encoding of the Delivery Mode field in the S_UNIDATA_REQUEST and S_EXPEDITED_UNIDATA_REQUEST primitives

A.2.2.28.3 TRANSMISSION-MODE Encoding for the S_UNIDATA_INDICATION and S_EXPEDITED_UNIDATA_INDICATION Primitives

Argument: TRANSMISSION-MODE
S_Primitives: S_UNIDATA_INDICATION, S_EXPEDITED_UNIDATA_INDICATION


ENCODING OF "TRANSMISSION MODE" FIELD:
S_UNIDATA_INDICATION AND S_EXPEDITED_UNIDATA_INDICATION
MSB
LSB
7
6
5
4
3
2
1
0
TRANSMISSION MODE

TRANSMISSION MODE
0000 = Not Used
0001 = ARQ
0010 = Non-ARQ (Broadcast)
0011 = Non-ARQ WITH ERRORS
0100 ....= to be defined

Figure A-30: 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 or Expedited U_PDU Argument with the TRANSMISSION-MODE argument. The TRANSMISSION-MODE argument in the S_UNIDATA_INDICATION and S_EXPEDITED_UNIDATA_INDICATION Primitives **shall** (1) be encoded as shown in Figure A-30.
[Note: The unused bits in this argument are allocated to the SOURCE SAP_ID argument encoding for both the S_UNIDATA_INDICATION and S_EXPEDITED_UNIDATA_INDICATION Primitives.]

A.2.2.28.4 Link-Type Encoding in S_Primitives

Argument: LINK TYPE
S_Primitives: S_HARD_LINK_ESTABLISH, S_HARD_LINK_ESTABLISHED, S_HARD_LINK_REJECTED, S_HARD_LINK_ACCEPT, S_HARD_LINK_INDICATION, S_HARD_LINK_REJECT, S_HARD_LINK_TERMINATED

A client uses the Link-Type argument to reserve partially or fully the capacity of the Hard Link. This argument can have three values:

- A value of 0 **shall** (1) indicate that the physical link to the specified node address is a Type 0 Hard Link. The Type 0 Hard Link must be maintained, but all clients connected to the two nodes can make use of the link capacity according to normal procedures, i.e. there is no bandwidth reservation.

- A value of 1 **shall** (2) indicate that the physical link to the specified node address is a Type 1 Hard Link. The Type 1 Hard Link must be maintained and traffic is only allowed between the requesting client and any of the clients on the remote Node, i.e., there is partial bandwidth reservation.

- A value of 2 indicates that the physical link to the specified node address must be maintained and traffic is only allowed between the requesting Client and the specific Client on the remote node specified by the remote SAP ID argument, i.e. full bandwidth reservation.

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:

1. Establishment and Termination of Hard Link Data Exchange Sessions
2. Exchange of Client Data

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 eight 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-31.

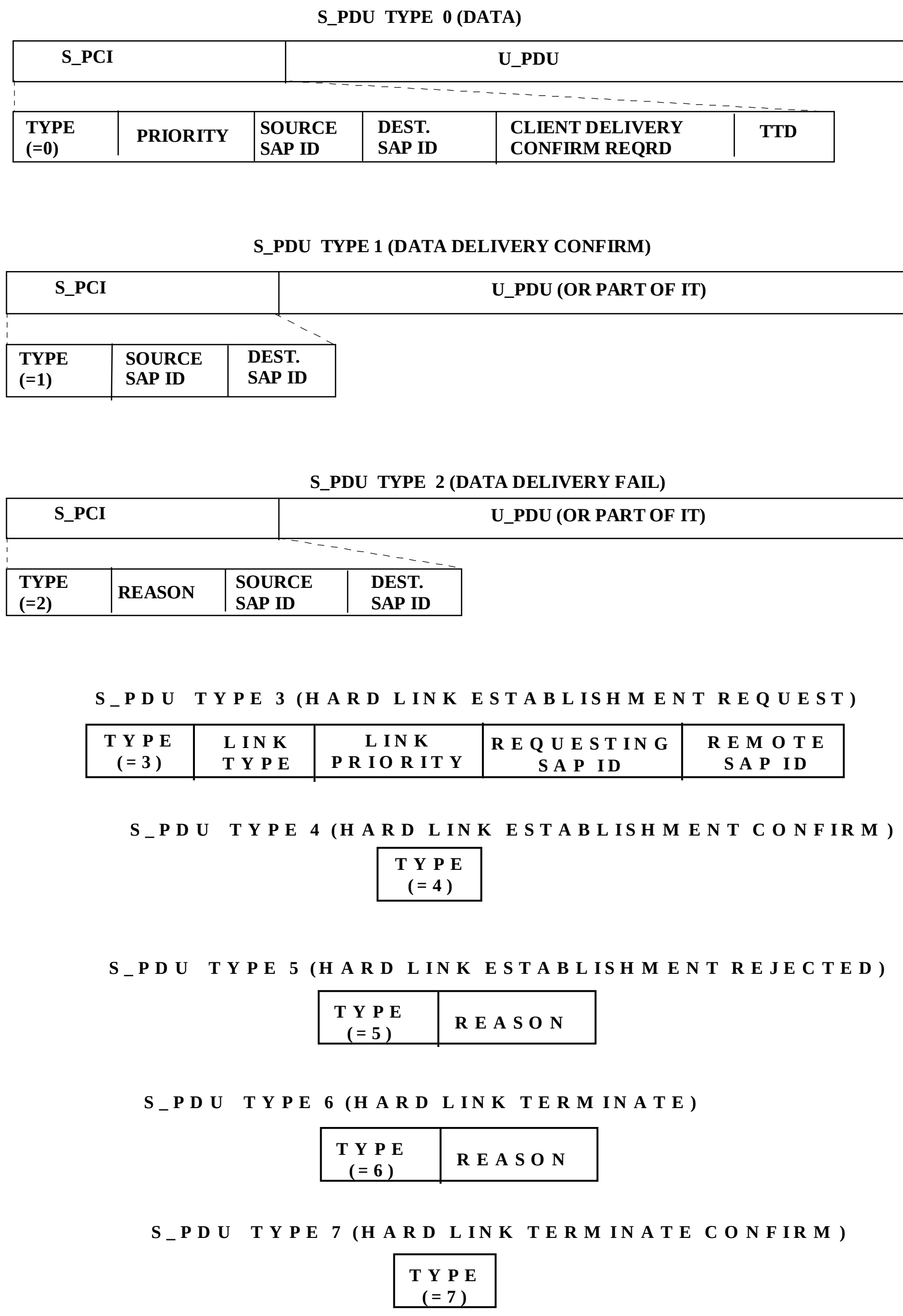

S_PDU TYPE 0 (DATA)
S_PCI
U_PDU
TYPE (=0)
PRIORITY
SOURCE SAP ID
DEST. SAP ID
CLIENT DELIVERY CONFIRM REQRD
TTD
S_PDU TYPE 1 (DATA DELIVERY CONFIRM)
S_PCI
U_PDU (OR PART OF IT)
TYPE (=1)
SOURCE SAP ID
DEST. SAP ID
S_PDU TYPE 2 (DATA DELIVERY FAIL)
S_PCI
U_PDU (OR PART OF IT)
TYPE (=2)
REASON
SOURCE SAP ID
DEST. SAP ID
S_PDU TYPE 3 (HARD LINK ESTABLISHMENT REQUEST)
TYPE (=3)
LINK TYPE
LINK PRIORITY
REQUESTING SAP ID
REMOTE SAP ID
S_PDU TYPE 4 (HARD LINK ESTABLISHMENT CONFIRM)
TYPE (=4)
S_PDU TYPE 5 (HARD LINK ESTABLISHMENT REJECTED)
TYPE (=5)
REASON
S_PDU TYPE 6 (HARD LINK TERMINATE)
TYPE (=6)
REASON
S_PDU TYPE 7 (HARD LINK TERMINATE CONFIRM)
TYPE (=7)

Figure A-31: 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 field value	S_PDU Name
0	DATA
1	DATA DELIVERY CONFIRM
2	DATA DELIVERY FAIL
3	HARD LINK ESTABLISHMENT REQUEST
4	HARD LINK ESTABLISHMENT CONFIRM
5	HARD LINK ESTABLISHMENT REJECTED
6	HARD LINK TERMINATE
7	HARD LINK TERMINATE CONFIRM

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.

A.3.1.1 DATA S_PDU

Type :

"0" = DATA S_PDU

Encoding :

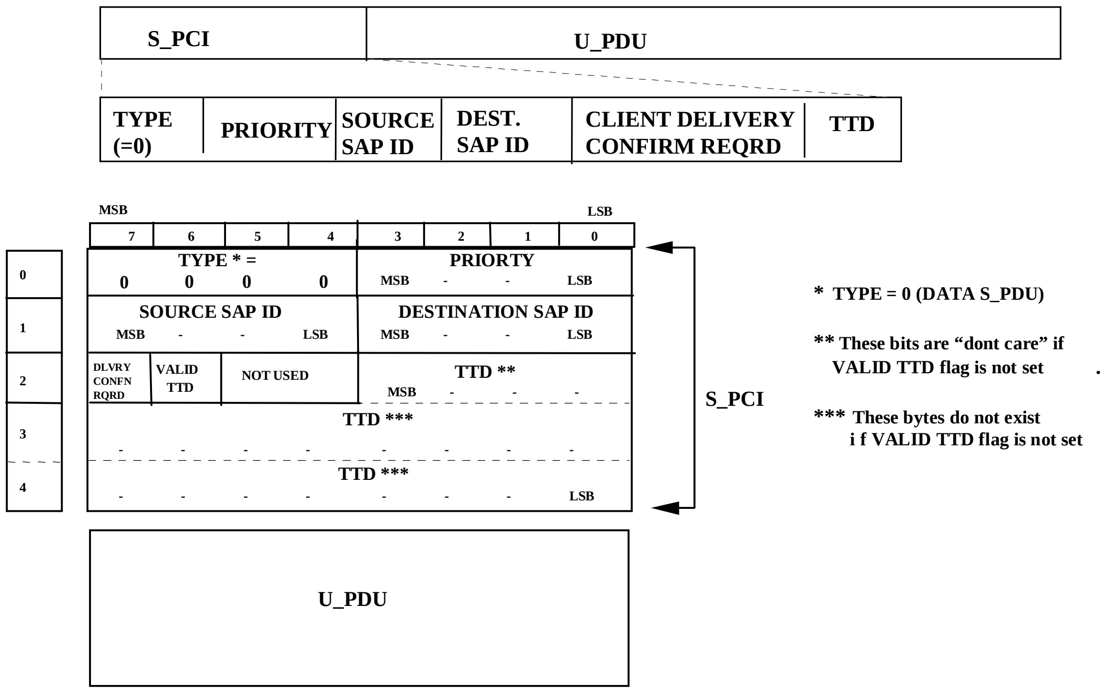


Figure A-32: Generic Encoding and Bit-Field Map of the DATA S_PDU

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-32 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** (14) be set according to the *Service Type* requested by the sending client during binding (see S_BIND_REQUEST primitive) or according to the *Delivery Mode* requested explicitly for this U_PDU (see S_UNIDATA_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[3] 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 Julian date modulo 16 part of the TTD **shall** (18) be mapped into the first four bits of the TTD field (i.e., bits 0-3 of byte 2 of the S_PDU).

[3] 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 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 :

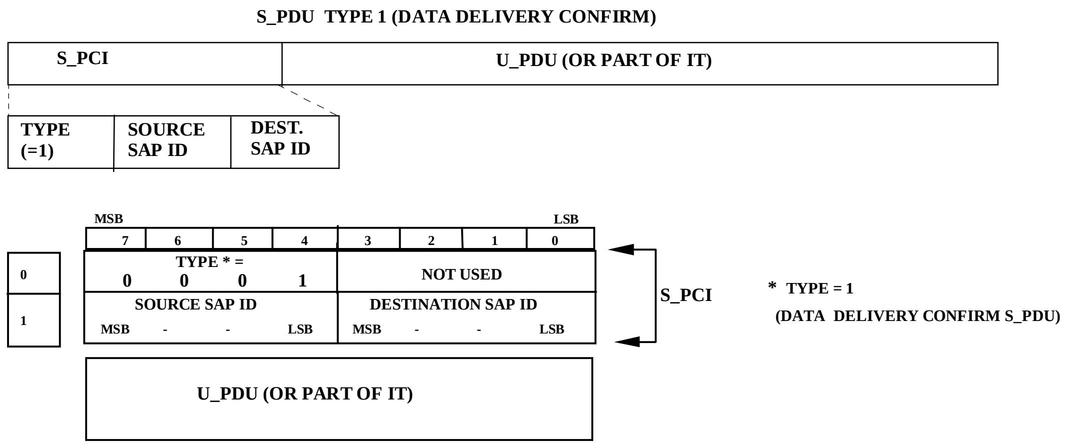


Figure A-33: Generic Encoding and Bit-Field Maps of the DATA DELIVERY CONFIRM S_PDU

Description :

The DATA DELIVERY CONFIRM S_PDU **shall** be (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 or a S_EXPEDITED_UNIDATA_REQUEST_CONFIRM in accordance with the data exchange protocol of Section A.3.2.4.

A.3.1.3 DATA DELIVERY FAIL S_PDU

Type :

“2” = DATA DELIVERY FAIL

Encoding :

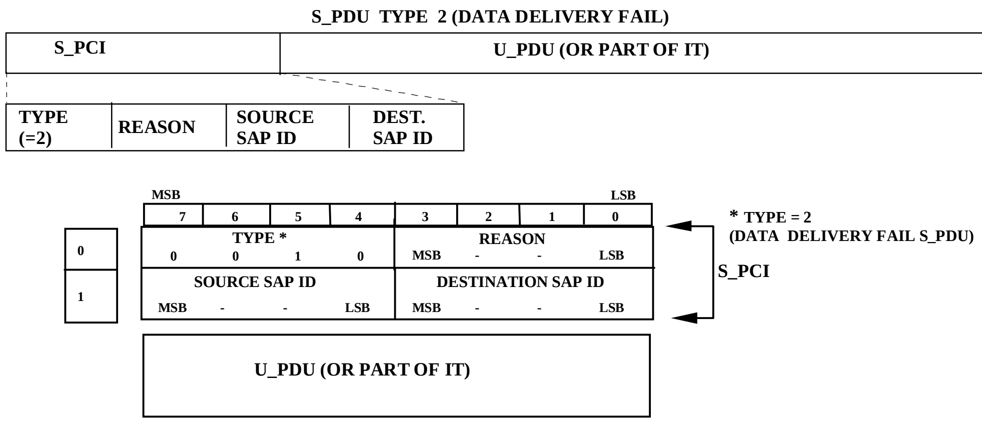


Figure A-34: Generic Encoding and Bit-Field Map of the DATA DELIVERY FAIL S_PDU

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 or a S_EXPEDITED_UNIDATA_REQUEST_REJECTED primitive, in accordance with the data exchange protocol of Section A.3.2.4.

A.3.1.4 HARD LINK ESTABLISHMENT REQUEST S_PDU

Type :

“3” = HARD LINK ESTABLISHMENT REQUEST

Encoding :

TYPE (=3)	LINK TYPE	LINK PRIORITY	REQUESTING SAP ID	REMOTE SAP ID

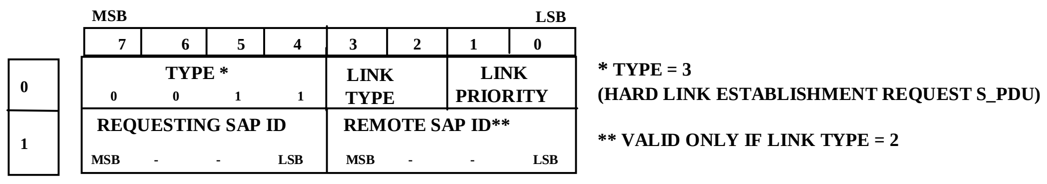


Figure A-35: Generic Encoding and Bit-Field Map of the HARD LINK ESTABLISHMENT REQUEST S_PDU

Description :

The HARD LINK ESTABLISHMENT REQUEST S_PDU **shall** (1) be transmitted by a Peer in response to a Client’s request for a Hard Link. Since the establishment of a Hard Link overrides the normal procedures of making Links based on the destinations of the queued U_PDUs (i.e., over Soft Link Data Exchange), it is important that both peers use a handshake procedure in order to confirm the successful Hard Link establishment

The first field of the S_PDU **shall** (2) be “TYPE”. It **shall** (3) be equal to 3 and identifies the S_PDU as being of type HARD LINK ESTABLISHMENT REQUEST.

The “LINK TYPE” and “LINK PRIORITY” fields **shall** (4) be equal in value to the corresponding arguments of the S_HARD_LINK_ESTABLISH Primitive submitted by the client to request the link.

The “REQUESTING SAP ID” field **shall** (5) be the SAP ID of the client that requested the Hard Link Establishment.

This “REMOTE SAP ID” field **shall** (6) be valid if and only if the “LINK TYPE” field has a value of 2, denoting a Type 2 Hard Link w/ Full-Bandwidth Reservation. The “REMOTE SAP ID” field **shall** (7) identify the single client connected to the remote node to and from which traffic is allowed for Hard Links w/ Full-Bandwidth Reservation. The REMOTE SAP ID field may take any implementation-dependent default value for Hard Links without Full Bandwidth Reservation.

A.3.1.5 HARD LINK ESTABLISHMENT CONFIRM S_PDU

Type :

“4” = HARD LINK ESTABLISHMENT CONFIRM

Encoding :

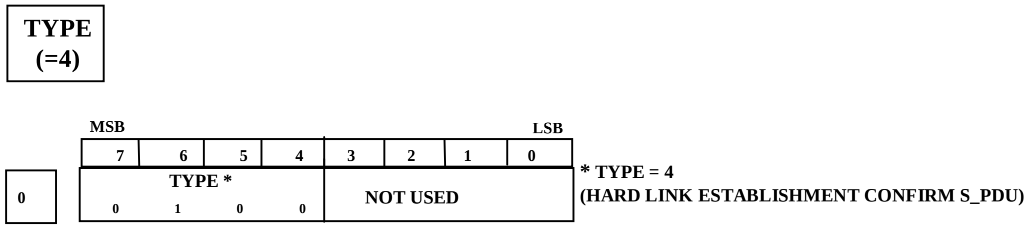


Figure A-36: Generic Encoding and Bit-Field Map of the HARD LINK ESTABLISHMENT CONFIRM S_PDU

Description :

The HARD LINK ESTABLISHMENT CONFIRM S_PDU **shall** (1) be transmitted as a positive response to the reception of a HARD LINK ESTABLISHMENT REQUEST S_PDU.

Its only field **shall** (2) be “TYPE”, which value **shall** (3) be equal to 4 and identifies the S_PDU as being of type HARD LINK ESTABLISHMENT CONFIRM.

The peer which receives this S_PDU **shall** (4) inform its appropriate client accordingly with a S_HARD_LINK_ESTABLISHED Primitive in accordance with the Hard Link Establishment Protocol specified in Section A.3.2.2.2.

A.3.1.6 HARD LINK ESTABLISHMENT REJECTED S_PDU

Type :

“5” = HARD LINK ESTABLISHMENT REJECTED

Encoding :

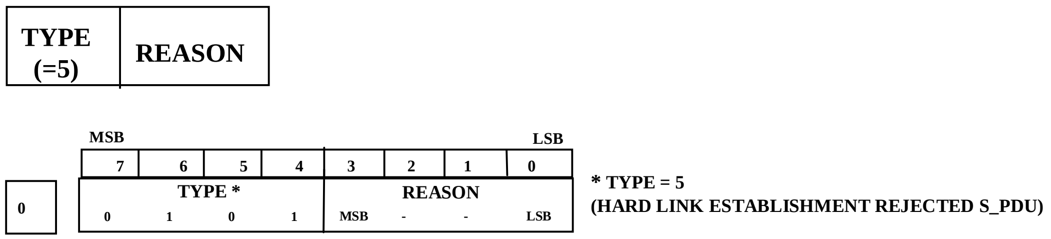


Figure A-37: Generic Encoding and Bit-Field Map of the HARD LINK ESTABLISHMENT REJECTED S_PDU

Description :

This S_PDU **shall** (1) be transmitted as a negative response to the reception of a HARD LINK ESTABLISHMENT REQUEST S_PDU.

The first field **shall** (2) be “TYPE” and its value **shall** (3) be equal to 5 to identify the S_PDU as being of type HARD LINK ESTABLISHMENT REJECTED.

The second field **shall** (4) be “REASON” and explains why the Hard Link request was rejected. The sublayer that receives this S_PDU should inform its appropriate client accordingly with a S_HARD_LINK_REJECTED Primitive in accordance with the Hard Link Establishment Protocol specified in Section A.3.2.2.2. The “REASON” field **shall**(5) take a value in the range 0-15. Hard Link reject reasons and their corresponding values **shall** (6) be as defined in the following table.

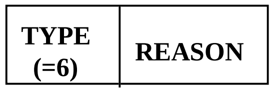
REASON	Field Value
unassigned	0
Remote-Node-Busy	1
Higher-Priority-Link-Existing	2
Remote-Node-Not-Responding	3
Destination SAP ID not bound	4
Reserved for future use	5-15

A.3.1.7 HARD LINK TERMINATE S_PDU

Type :

“6” = HARD LINK TERMINATE

Encoding :



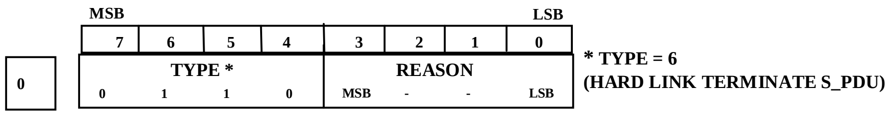


Figure A-38: Generic Encoding and Bit-Field Map of the HARD LINK TERMINATE S_PDU

Description :

Under normal circumstances a Hard Link is terminated at the request of the Client which originated it or as a result of a request by another Client to establish a higher priority Hard Link. Either of the two Peer sublayers involved in a Hard Link session and that wishes to terminate the Hard Link **shall** (1) transmit a HARD LINK TERMINATE S_PDU to request termination of the Hard Link.

The first four-bit field **shall** (2) be “TYPE” and its value **shall** (3) be set equal to 6 to identify the S_PDU as being of type HARD LINK TERMINATE.

The second four-bit field **shall** (4) be “REASON” and explains why the Hard Link is being terminated. Hard Link terminate reasons and their corresponding values **shall** (5) be as defined in the following table.

Reason	Value
unassigned	0
Client request	1
Higher priority link requested	2
reserved	3
Destination SAP ID unbound	4
Reserved for future use	5-15

In order to ensure a graceful termination of the Hard Link, the Peer which sent the HARD LINK TERMINATE must await a TIMEOUT period for confirmation of its Peer before it declares the Link as terminated. This TIMEOUT period **shall** (6) be configurable by the protocol implementation.

A.3.1.7 HARD LINK TERMINATE CONFIRM S_PDU

Type :

"7" = HARD LINK TERMINATE CONFIRM

Encoding :

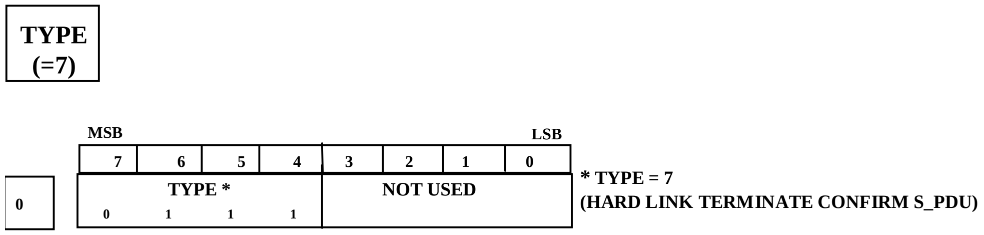


Figure A-39: Generic Encoding and Bit-Field Map of the HARD LINK TERMINATE CONFIRM S_PDU

Description :

The HARD LINK TERMINATE CONFIRM S_PDU **shall** (1) be transmitted in response to the reception of a HARD LINK TERMINATE S_PDU.

The first four-bit field of this S_PDU **shall** (2) be "TYPE". A value of 7 **shall** (3) identify that the S_PDU is of type HARD LINK TERMINATE CONFIRM.

The second four-bit field of this S_PDU **shall** (3) be not used in this implementation of the protocol. The values of these bits may be implementation dependent.

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 Hard 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

In the absence of a hard link request by a client, 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 Protocol for Establishing a Soft Link Data Exchange Session

In contrast with the establishment of a Hard Link 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.3.2.1.2 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.3.2.2 Hard Link Data Exchange Session

The rules governing establishment and termination of Hard Links are straightforward but complicated by the fact that new Hard Link requests could be satisfied (at least in part) by an existing Hard Link. Comparison and evaluation factors for Hard Links include the ranks of the clients, the link priorities, the sets of source and destination nodes, and the Hard-Link types. In particular, various service models could be specified in this STANAG for the management of multiple Hard Links of Types 0 or Type 1 simultaneously, but with different levels of protocol complexity to track the potentially overlapping and independent requests from multiple clients. Note that, since Type 2 Hard Links reserve the full bandwidth and use of a physical link for two specific bound clients, the subnetwork cannot support multiple Type 2 Hard Links with any assumed service model.

This STANAG assumes a simple model for the management of Hard Links based on maintenance of at most a single Hard Link between nodes, while still allowing Type 0 and Type 1 Hard Links between the nodes to be shared by other clients using Soft Link Data Exchange. This management model satisfies the following requirements:

- a node's sublayer **shall** (1) maintain at most one Hard Link at any time;
- a sublayer **shall** (2) accept a Hard Link request when no Hard Link currently exists;
- the comparative precedence of new requests and any existing hard link **shall** (3) be evaluated in accordance with section A.3.2.2.1 to determine if the new request can be accepted or rejected by the sublayer;
- requests of higher precedence **shall** (4) be accepted and will result in the termination of an existing Hard Link;
- an existing Hard Link of higher precedence **shall** (5) result in the rejection of the request;
- if an existing Type 0 Hard Link can satisfy a request that has been rejected, the sublayer **shall** (6) note this as the reason for rejecting the request.; the requesting client may then submit data for transmission using a Soft Link Data Exchange Session.

[Note: While this model has some limitations, notably that clients sharing a Hard Link with the originator will lose it when the originator terminates the link, it supports the essential service characteristics desired, and presents a simpler protocol than others that were considered.]

Unless noted otherwise, any data structures and variables used to manage Hard Link establishment and termination are implementation dependent and beyond the scope of this STANAG.

Further requirements controlling the establishment and termination of Hard Links are specified below.

A.3.2.2.1 Priority and Precedence Rules for Hard Links

Establishment and Termination of Hard Links **shall** (1) be controlled in accordance with the following set of precedence rules:

a) A Hard Link request for a client with greater rank **shall** (2) take precedence over an existing or requested Hard Link established for a client of lower rank, regardless of other factors.

b) A Hard Link request of greater priority **shall** (3) take precedence over an existing or requested hard link of lower priority, regardless of other factors.

c) For Hard Links of equal priority and rank, and with different sets of source and destination nodes, the Hard Link request processed first by the Subnetwork Interface Sublayer (i.e., the Hard Link currently established) **shall** (4) take precedence.

d) For Hard Links (i.e., requests and existing hard links) from clients of equal priority and rank, and with equal sets of source and destination nodes:

- a Hard Link with greater Link Type value **shall** (5) take precedence over one with lower value;
- an existing Hard Link **shall** (6) take precedence over subsequent Hard Link requests of equal Link Type.

A.3.2.2.2 Protocol for Establishing a Hard Link Data Exchange Session

Upon receiving a S_HARD_LINK_ESTABLISH Primitive from a client, the Subnetwork Interface Sublayer **shall** (1) first check that it can accept the request from the client in accordance with the precedence and priority rules of Section A.3.2.2.1.

If the Hard Link request is of lower precedence than any existing Hard Link, then the establishment protocol proceeds as follows:

- the request **shall** (2) be denied by the Subnetwork Interface Sublayer,
- the sublayer **shall** (3) issue a S_HARD_LINK_REJECTED Primitive to the requesting client with REASON = “Higher-Priority-Link-Existing”, and
- the sublayer **shall** (4) terminate the Hard Link establishment protocol.

Otherwise, if a Type 0 Hard Link request is of the same priority, same client-rank, and with the same set of source and destination nodes as an existing Hard Link, then the establishment protocol proceeds as follows:

- the Subnetwork Interface Sublayer **shall** (5) reject the Type 0 Hard Link request with the REASON = “Requested Type 0 Hard Link Exists”; a client receiving this rejection may submit data requests for transmission using a Soft-Link Data Exchange Session to the remote peer;
- the sublayer **shall** (6) take no further action to establish or change the status of the existing Type 0 Hard Link (Note: since the sublayer has already determined that the existing link satisfies the requirements of the request), and; the sublayer **shall** (7) terminate the Hard Link establishment protocol.

Otherwise, the establishment protocol proceeds in accordance with the following requirements.

If the Subnetwork Interface Sublayer can accept the Hard Link request it **shall** (8) first terminate any existing Hard Link of lower precedence using the peer-to-peer communication protocol for terminating an existing hard link specified in Section A.3.2.2.3.

The Subnetwork Interface Sublayer then **shall** (9) request the Channel Access Sublayer to make a physical link to the node specified by the client, following procedure for making the physical link specified in Annex B.

After the physical link has been made, the caller's Subnetwork Interface Sublayer **shall** (10) send a "HARD LINK ESTABLISHMENT REQUEST" (TYPE 3) S_PDU to its called peer at the remote node. To ensure that the S_PDU will overtake all routine DATA S_PDUs which may be queued and in various stages of processing by the lower sublayers, the "HARD LINK ESTABLISHMENT REQUEST" S_PDU **shall** (11) be sent to the Channel Access Sublayer using a C_EXPEDITED_UNIDATA_REQUEST Primitive and use the subnetwork's expedited data service.

After it sends the "HARD LINK ESTABLISHMENT REQUEST" (TYPE 3) S_PDU, the caller's Subnetwork Interface Sublayer **shall** (12) wait a configurable time-out period for a response from the called peer, and proceed as follows:

- during the waiting-period for the response,

    - if the caller's sublayer receives a HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU from the called peer, the sublayer **shall** (13) notify the requesting client that the Hard Link request has failed by sending the client an S_HARD_LINK_REJECTED Primitive to the requesting client with the REASON field of the Primitive set to the corresponding value received in the S_PDU's REASON field,

    - if the caller's sublayer receives a HARD LINK ESTABLISHMENT CONFIRM" (TYPE 4) S_PDU from the called peer, the sublayer **shall** (14) notify the requesting client that the Hard Link request has succeeded by sending the client an S_HARD_LINK_ESTABLISHED Primitive;

- otherwise, if the waiting-period for the response expires without receipt of a valid response from called node, the caller's sublayer **shall** (15) notify the requesting client that the Hard Link request has failed by sending the client an S_HARD_LINK_REJECTED Primitive to the requesting client with REASON = "Remote-Node-Not Responding".

The caller's establishment protocol **shall** (16) terminate on receipt during the waiting of a valid response from the called node and notification of the client, or on expiration of the waiting period.

For the called Subnetwork Access Sublayer, the Hard Link establishment protocol **shall** (17) be initiated on receipt of a "HARD LINK ESTABLISHMENT REQUEST" (TYPE 3) S_PDU, and proceeds as follows:

- if no client is bound to the called SAP ID and the caller's request is for a Type 2 Hard Link, then the called sublayer **shall** (18) reject the request, send a "HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU with REASON = "Destination SAP ID not bound" to the caller, and terminate the establishment protocol;

- otherwise, the called sublayer **shall** (19) evaluate the precedence of the caller's request in accordance with the precedence and priority rules of Section A.3.2.2.1, using as the client rank either a configurable default rank for the called SAP_ID for Type 0 and Type 1 Hard Link requests, or the actual rank of the bound client with the called SAP_ID for Type 2 Hard Link requests.

- If the caller's request cannot be accepted by the called peer, a "HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU **shall** (21) be sent to the calling peer, with the Reason field set as follows:

  - REASON = "Remote-Node-Busy" if the reason for rejection was the existence of an existing Hard Link of equal rank and priority, or,

  - REASON= "Higher-Priority Link Existing" if the reason for rejection was the existing of a Hard Link with higher priority or rank.

- If the caller's Hard Link request can be accepted and the request is not a Type 2 Hard Link request, the called sublayer **shall** (22) send a "HARD LINK ESTABLISHMENT CONFIRM" (TYPE 4) S_PDU to the caller sublayer, and terminate the protocol;

- otherwise, the request is for a Type 2 Hard Link and the called sublayer **shall** (23) send a S_HARD_LINK_INDICATION Primitive to the requested client, and wait for a configurable maximum time-out period for a response:

  - if the called sublayer receives a S_HARD_LINK_ACCEPT Primitive from the requested client prior to the expiration of the timeout, then the called sublayer **shall** (24) send a "HARD LINK ESTABLISHMENT CONFIRM" (TYPE 4) S_PDU to the calling sublayer, and terminate the protocol;

  - otherwise, the called sublayer **shall** (24) send a "HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU to the caller sublayer, and terminate the protocol.

- Whenever sent, either the TYPE 4 (HARD LINK ESTABLISHMENT CONFIRM) S_PDU or the TYPE 5 (HARD LINK ESTABLISHMENT REJECTED) S_PDU **shall** (25) be sent to the calling sublayer using the Expedited Data Service provided by lower sublayers in the profile.

The procedures for establishing a hard link from the perspective of both the calling and called peers are depicted in Figure A-40(a) and Figure A-40(b) as a possible implementation that meets the stated requirements. This STANAG acknowledges that other implementations may exist that also meet the stated requirements.

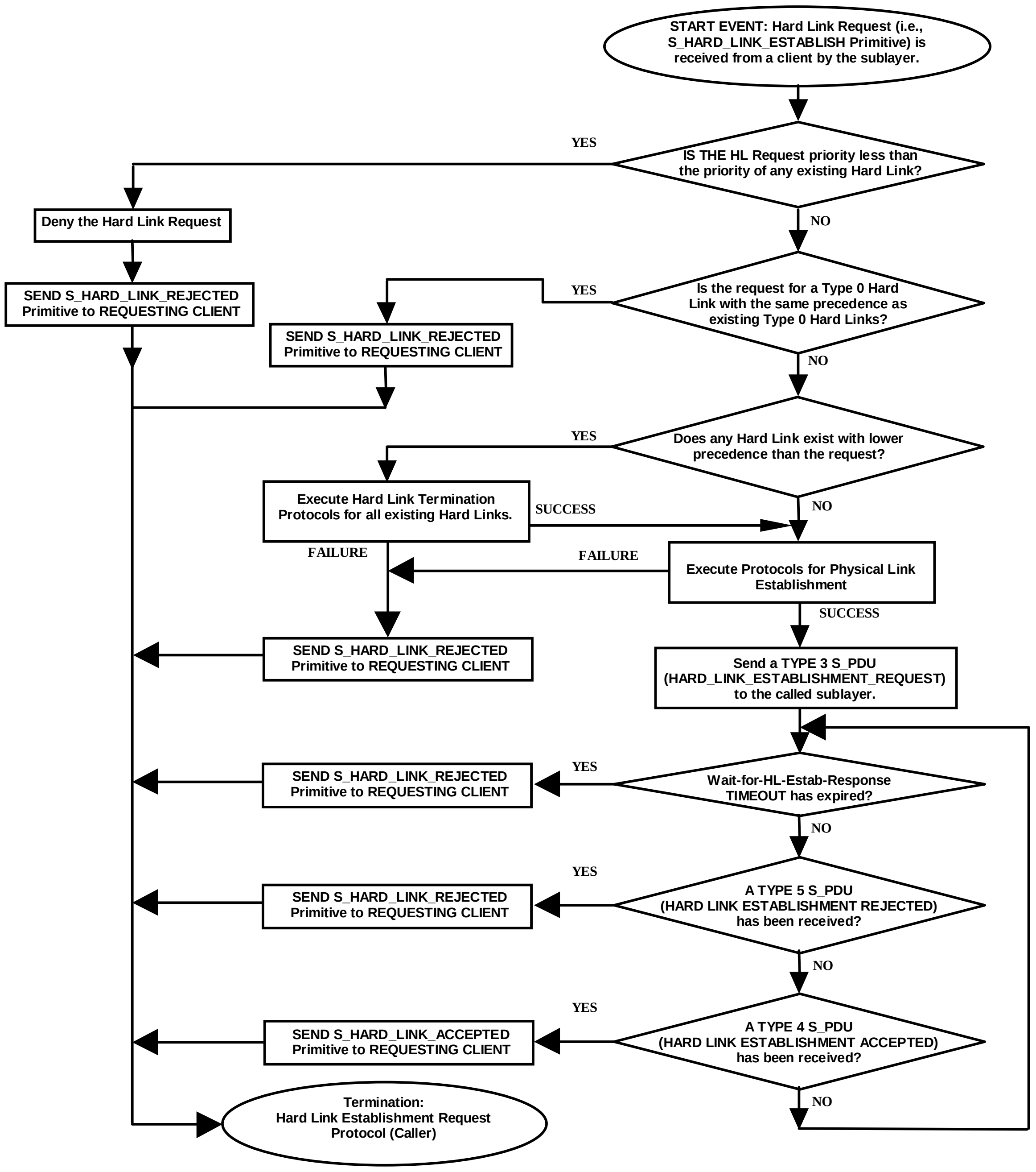

START EVENT: Hard Link Request (i.e., S_HARD_LINK_ESTABLISH Primitive) is received from a client by the sublayer.
IS THE HL Request priority less than the priority of any existing Hard Link?
YES
NO
Deny the Hard Link Request
SEND S_HARD_LINK_REJECTED Primitive to REQUESTING CLIENT
Is the request for a Type 0 Hard Link with the same precedence as existing Type 0 Hard Links?
YES
NO
SEND S_HARD_LINK_REJECTED Primitive to REQUESTING CLIENT
Does any Hard Link exist with lower precedence than the request?
YES
NO
Execute Hard Link Termination Protocols for all existing Hard Links.
SUCCESS
FAILURE
FAILURE
Execute Protocols for Physical Link Establishment
SUCCESS
SEND S_HARD_LINK_REJECTED Primitive to REQUESTING CLIENT
Send a TYPE 3 S_PDU (HARD_LINK_ESTABLISHMENT_REQUEST) to the called sublayer.
Wait-for-HL-Estab-Response TIMEOUT has expired?
YES
NO
SEND S_HARD_LINK_REJECTED Primitive to REQUESTING CLIENT
A TYPE 5 S_PDU (HARD LINK ESTABLISHMENT REJECTED) has been received?
YES
NO
SEND S_HARD_LINK_REJECTED Primitive to REQUESTING CLIENT
A TYPE 4 S_PDU (HARD LINK ESTABLISHMENT ACCEPTED) has been received?
YES
NO
SEND S_HARD_LINK_ACCEPTED Primitive to REQUESTING CLIENT
Termination: Hard Link Establishment Request Protocol (Caller)

Figure A-40 (a): Procedures for Establishing a Hard Link: CALLER PEER

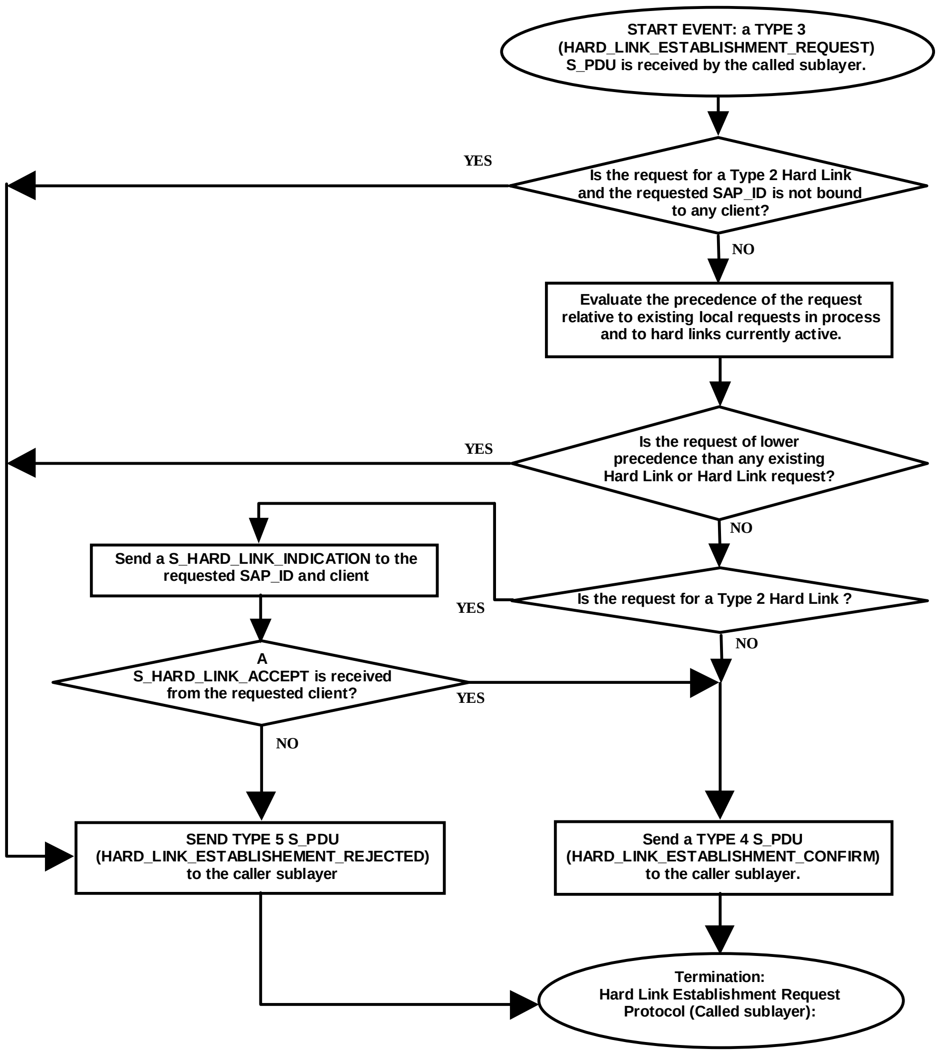

START EVENT: a TYPE 3 (HARD_LINK_ESTABLISHMENT_REQUEST) S_PDU is received by the called sublayer.
Is the request for a Type 2 Hard Link and the requested SAP_ID is not bound to any client?
YES
NO
Evaluate the precedence of the request relative to existing local requests in process and to hard links currently active.
Is the request of lower precedence than any existing Hard Link or Hard Link request?
YES
NO
Send a S_HARD_LINK_INDICATION to the requested SAP_ID and client
Is the request for a Type 2 Hard Link ?
YES
NO
A S_HARD_LINK_ACCEPT is received from the requested client?
YES
NO
SEND TYPE 5 S_PDU (HARD_LINK_ESTABLISHEMENT_REJECTED) to the caller sublayer
Send a TYPE 4 S_PDU (HARD_LINK_ESTABLISHMENT_CONFIRM) to the caller sublayer.
Termination: Hard Link Establishment Request Protocol (Called sublayer):

Figure A-40 (b): Procedures for Establishing a Hard Link: CALLED PEER

A.3.2.2.3 <u>Protocol for Terminating a Hard Link Data Exchange Session</u>

The termination of an existing Hard Link can be initiated by either of the two peer sublayers connected by the link. Normally the Hard Link will be terminated by the calling sublayer at the request of the client who initiated it, or by either of the sublayers if it receives a Hard Link request of higher precedence from one of its other clients. Requirements for the Hard Link termination protocol are defined below.

The Hard Link termination protocol **shall** (1) be initiated when any of the following conditions are met:

- a calling sublayer receives a S_HARD_LINK_TERMINATE Primitive from the client that originated an existing hard link of any type,
- a called sublayer receives a S_HARD_LINK_TERMINATE Primitive from its attached client involved in an existing Type 2 Hard Link,

- either the calling or called sublayer receives from a client a S_HARD_LINK_ESTABLISH Primitive that is of higher precedence than any existing Hard Link.

Any sublayer that must terminate a Hard Link for any of the specified conditions **shall** (2) send a "HARD LINK TERMINATE" (TYPE 6) S_PDU to its peer sublayer.

A sublayer that receives a "HARD LINK TERMINATE" (TYPE 6) S_PDU from its peer **shall** (3) immediately respond with a "HARD LINK TERMINATE CONFIRM" (TYPE 7) S_PDU, declare the Hard Link as terminated, and send a S_HARD_LINK_TERMINATED Primitive to all clients using the Hard Link.

After sending the HARD LINK TERMINATE" (TYPE 6) S_PDU, the initiating sublayer **shall** (4) wait for a response for a configurable maximum time-out period, and proceed.

If the timeout-period expires without receipt by the initiating sublayer of a "HARD LINK TERMINATE CONFIRM" (TYPE 7) S_PDU, the sublayer **shall** (5) send a S_HARD_LINK_TERMINATED Primitive to all clients using the Hard Link, with the REASON field set equal to "Remote Node Not Responding (timeout)".

To ensure that any S_PDU used for the termination protocol will overtake all routine DATA S_PDUs that may be queued and in various stages of processing by the lower sublayers, the termination protocol uses the subnetworks's Expedited Data Service. In particular, the "HARD LINK TERMINATE" (TYPE 6) S_PDU and "HARD LINK TERMINATE CONFIRM" (TYPE 7) S_PDU **shall** (6) be sent to the Channel Access Sublayer using a C_EXPEDITED_UNIDATA_REQUEST Primitive.
.
After termination of the Hard Link with a subnetwork client, the Physical Link between the nodes may need to be broken. Normally the breaking of the Physical Link is left to the peer which requested the termination of the Hard Link session. The reason for this is that this peer may want to start another session using the existing Physical Link in which case breaking and making procedures may be avoided. The procedures for breaking a Physical Link are specified in Annex B.

The nominal procedures for Terminating a Hard Link for both the Requesting and Responding Peers are shown in Figure A-41. This STANAG acknowledges that other implementations may exist that meet the requirements stated above.

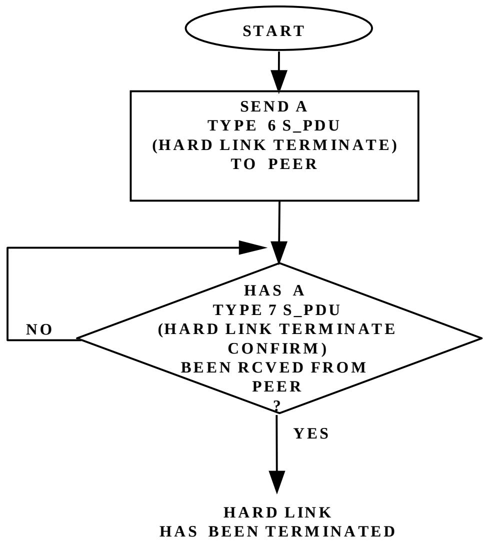

START
SEND A TYPE 6 S_PDU (HARD LINK TERMINATE) TO PEER
HAS A TYPE 7 S_PDU (HARD LINK TERMINATE CONFIRM) BEEN RCVED FROM PEER ?
NO
YES
HARD LINK HAS BEEN TERMINATED

Figure A-41 (a): Procedures for Terminating a Hard Link: REQUESTING PEER

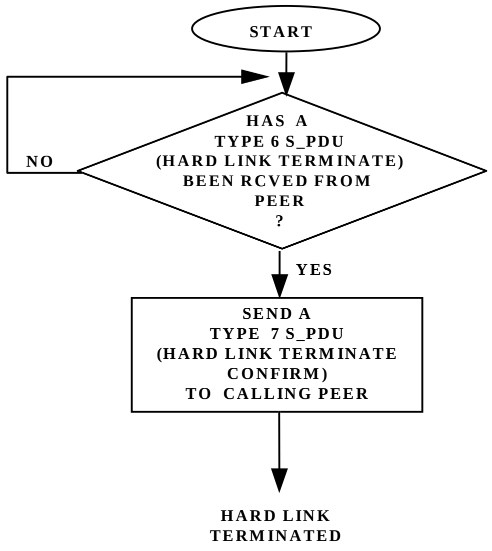

START
HAS A TYPE 6 S_PDU (HARD LINK TERMINATE) BEEN RCVED FROM PEER ?
NO
YES
SEND A TYPE 7 S_PDU (HARD LINK TERMINATE CONFIRM) TO CALLING PEER
HARD LINK TERMINATED

Figure A-42 (b): Procedures for Terminating a Hard Link: RESPONDING PEER

Apart from the procedures above, a sublayer **shall** (7) unilaterally declare a Hard Link as terminated if at any time it is informed by the Channel Access Sublayer that the physical link has been permanently broken. In this case, the sublayer **shall** (8) send a S_HARD_LINK_TERMINATED Primitive to all clients using the Hard Link, with the REASON field set equal to “Physical Link Broken”.

A.3.2.3 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.

When a Broadcast Data Exchange Session is first established the 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".

A.3.2.4 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:

- if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive the sublayer **shall** (4) send a S_UNIDATA_REQUEST_REJECT Primitive to the client;
- otherwise, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive, the sublayer **shall** (5) send a S_EXPEDITED_UNIDATA_REQUEST_REJECT Primitive to the client;
- for either form of the reject primitive, the REASON field **shall** (6) 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,:

  - if the encoded U_PDU was submitted by a client using a S_UNIDATA_REQUEST Primitive, then 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;

  - if the encoded U_PDU was submitted by a client using a S_EXPEDITED_UNIDATA_REQUEST Primitive, then the sublayer **shall** (10) encode the S_PDU as a C_EXPEDITED_UNIDATA_REQUEST Primitive;

- 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 Primitive or a C_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive prior to the end of the waiting time, the sublayer **shall** (13) send to the client either a S_UNIDATA_REQUEST_CONFIRM Primitive or S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive, respectively, where the type of C_Primitive expected and S_Primitive sent corresponds to the type of U_PDU delivery service requested;

  - otherwise, if the sublayer receives a C_UNIDATA_REQUEST_REJECT Primitive or a C_EXPEDITED_UNIDATA_REQUEST_REJECT Primitive prior to the end of the waiting time, the sublayer **shall** (14) send to the client a either a S_UNIDATA_REQUEST_REJECT Primitive or S_EXPEDITED_UNIDATA_REQUEST_CONFIRM REJECT, respectively, where the type of C_Primitive expected and S_Primitive sent corresponds to the type of U_PDU delivery service requested;

  - 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 either a S_UNIDATA_REQUEST_REJECT Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive, or a S_EXPEDITED_UNIDATA_REQUEST_REJECT Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive; for either reject S_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., either a C_UNIDATA_REQUEST_CONFIRM Primitive or a C_EXPEDITED_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 either a S_UNIDATA_REQUEST_CONFIRM Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive, or a S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST 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_REJECT Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive or a S_EXPEDITED_UNIDATA_REQUEST_REJECT Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive; for either form of the primitive, the REASON field shall be taken 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 either a S_UNIDATA_REQUEST_REJECT Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive, or a S_EXPEDITED_UNIDATA_REQUEST_REJECT Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive; for either 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.

  [Note: in accordance with the interface between the Subnetwork Interface and Channel Access sublayers, there is no explicit indication that the S_PDU is a "normal" or an "expedited" one. Whether the S_PDU is a "normal" or an "expedited" S_PDU is determined by the whether the S_PDU is encoded within a C_UNIDATA_INDICATION Primitives or a C_EXPEDITED_UNIDATA_INDICATION Primitives, respectively.]

- 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,

  - if the DATA (TYPE 0) S_PDU was encoded within a C_UNIDATA_INDICATION Primitive, 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 DATA (TYPE 0) S_PDU was encoded within a C_EXPEDITED_UNIDATA_INDICATION Primitive, the sublayer **shall** (24) deliver the extracted U_PDU to the destination client bound to Destination SAP_ID using a S_EXPEDITED_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; [Note: implementation-dependent methods may be used to provide additional determination that the client data was successfully delivered prior to sending the “DATA DELIVERY CONFIRM” (TYPE 1) S_PDU.],

  - 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. [Note: implementation-dependent methods may be used to provide additional determination that the client data was unsuccessfully delivered prior to sending the “DATA DELIVERY FAIL” (TYPE 1) S_PDU.]

- 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-43(a) and Figure A-43(b). This STANAG acknowledges that other implementations may satisfy the requirements stated above. It should be noted that, as shown in these figures, there is no explicit indication that the S_PDU is a “normal” or an “expedited” one. The reason for this is that the underlying sublayers are expected to treat Expedited S_PDUs differently and implicitly pass the information to the receiving peer by (for example) delivering Expedited S_PDUs as C_EXPEDITED_UNIDATA_INDICATION Primitives rather than normal C_UNIDATA_INDICATION Primitives.

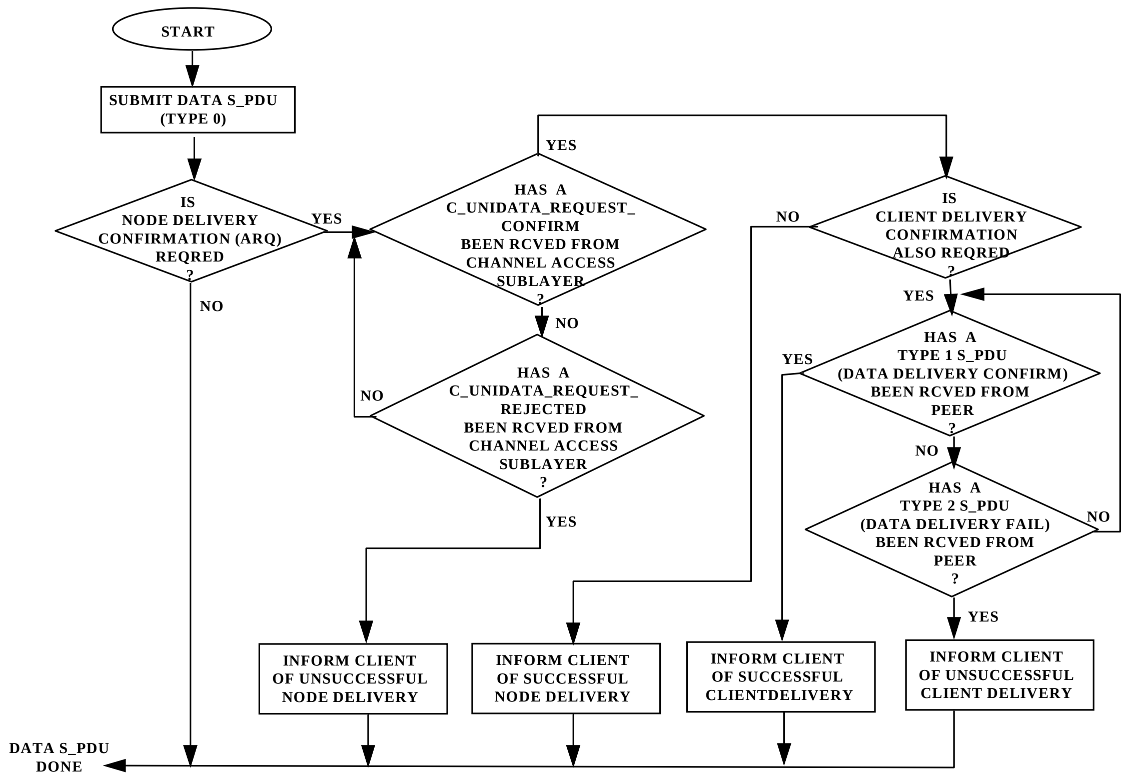

START
SUBMIT DATA S_PDU (TYPE 0)
IS NODE DELIVERY CONFIRMATION (ARQ) REQRED ?
YES
NO
HAS A C_UNIDATA_REQUEST_ CONFIRM BEEN RCVED FROM CHANNEL ACCESS SUBLAYER ?
YES
NO
HAS A C_UNIDATA_REQUEST_ REJECTED BEEN RCVED FROM CHANNEL ACCESS SUBLAYER ?
NO
YES
IS CLIENT DELIVERY CONFIRMATION ALSO REQRED ?
NO
YES
HAS A TYPE 1 S_PDU (DATA DELIVERY CONFIRM) BEEN RCVED FROM PEER ?
YES
NO
HAS A TYPE 2 S_PDU (DATA DELIVERY FAIL) BEEN RCVED FROM PEER ?
NO
YES
INFORM CLIENT OF UNSUCCESSFUL NODE DELIVERY
INFORM CLIENT OF SUCCESSFUL NODE DELIVERY
INFORM CLIENT OF SUCCESSFUL CLIENTDELIVERY
INFORM CLIENT OF UNSUCCESSFUL CLIENT DELIVERY
DATA S_PDU DONE

Figure A-43 (a): Data Exchange Procedures: SENDING PEER

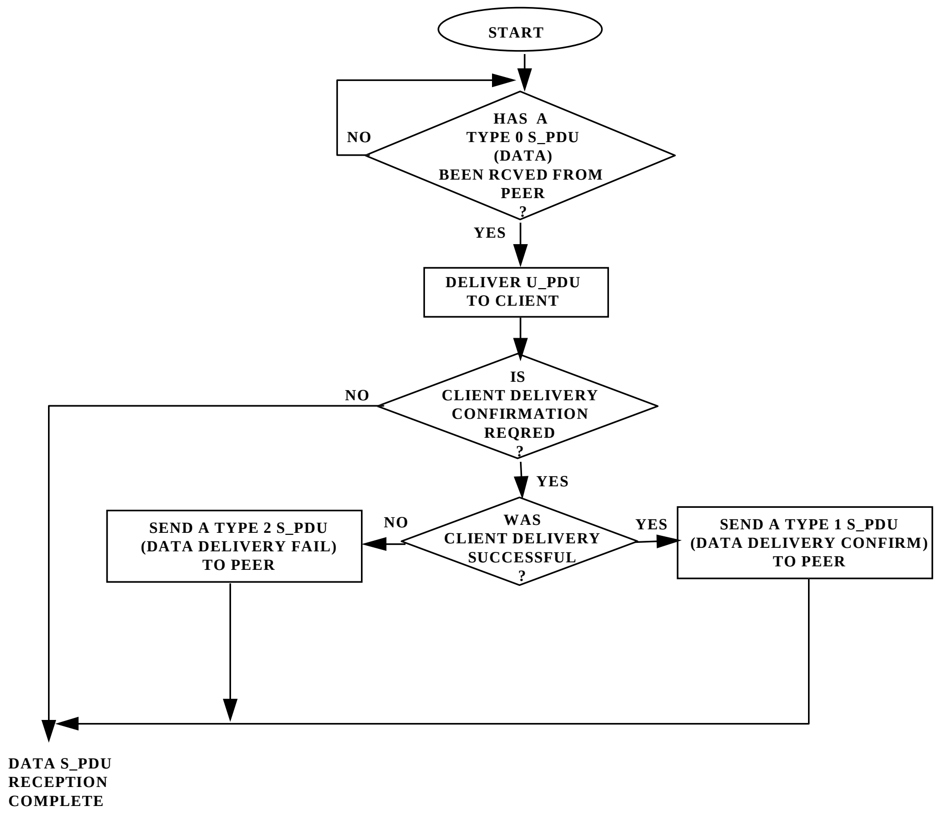

START
HAS A TYPE 0 S_PDU (DATA) BEEN RCVED FROM PEER ?
NO
YES
DELIVER U_PDU TO CLIENT
IS CLIENT DELIVERY CONFIRMATION REQRED ?
NO
YES
WAS CLIENT DELIVERY SUCCESSFUL ?
NO
YES
SEND A TYPE 2 S_PDU (DATA DELIVERY FAIL) TO PEER
SEND A TYPE 1 S_PDU (DATA DELIVERY CONFIRM) TO PEER
DATA S_PDU RECEPTION COMPLETE

Figure A-43 (b): Data Exchange Procedures: RECEIVING PEER