ANNEX S SIS ACCESS PROTOCOL (Mandatory)

This annex defines a protocol for clients to access the STANAG 5066 Subnet Interface Service. This enables multiple independent clients to easily share and multiplex over a single channel.

S.1. Accessing the Subnet Interface Service

S.1.1. Changes in This Edition

The functional differences between this specification and Edition 3 are set out in Section S.7.

This is a new annex, taking Edition 3 elements from Annex A and Annex F. It brings this core STANAG 5066 protocol into a single specification to improve modularity and clarity. Key components:

1. Service Encoding. The encoding of service primitives is only used by the SIS Access Protocol, so is specified in this Annex. In Edition 3, it was specified as Section A.3. It is therefore now specified in Annex S as part of the SIS Access Protocol specification.
2. Local Services (S_SUBNET_AVAILABILITY; S_DATA_FLOW_ON/OFF; S_MANAGEMENT_MSG; S_KEEPALIVE) are specified in this Annex. In Edition 3, they were specified in Section A.2.
3. The protocol structure of this document was specified as F.16 in Edition 3.

The following services specified in Edition 3 are optional and deprecated in Edition 4 Annex A (SIS). Their use in SIS Access Protocol is specified as follows:

1. Hard Links. These are now deprecated and encoding is specified in Section S.6.1.
2. Expedited data. This service is now deprecated and encoding is specified in Section S.6.2.

S.1.2. **Overview**

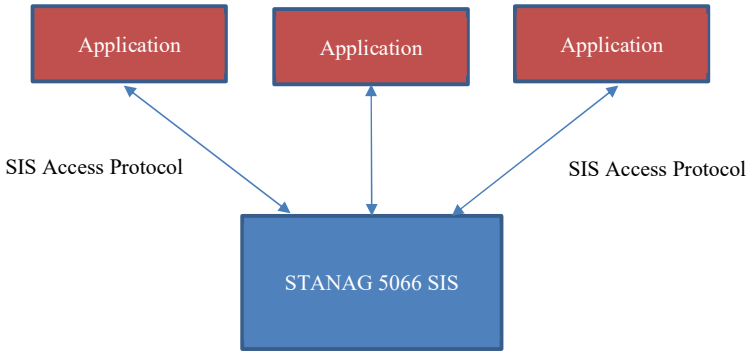


Figure S-1: SIS Access Protocol Model

A client-server relationship can be used to govern the interaction between the HF subnetwork and the users of the subnetwork, as shown in Figure S-1~~Figure S-1~~. The users (clients) request the services provided by the HF subnetwork (server). The service provided by the server is application independent and common to all clients irrespective of the task they may perform.

Clients are attached to the Subnetwork Interface Sublayer at Subnetwork Access Points (SAPs). There can be multiple clients simultaneously attached to the Subnetwork Interface Sublayer. Each SAP is identified by its SAP Identifier (SAP ID)1. The SAP ID is a number in the range 0-15; hence there can be a maximum of 16 clients attached to the Subnetwork Interface Sublayer of a single node.

The SIS Access protocol specified in this annex allows a client to access the Subnet Interface Service. There can be one client for each SAP, which enables multiplexing of multiple independent clients.

The SIS service specified in Annex A allows clients to access the SIS service by any mechanism. The mechanism specified in this annex is mandatory, to enable independent interoperable integration between STANAG 5066 severs and applications operating over STANAG 5066.

S.2. Mapping onto TCP

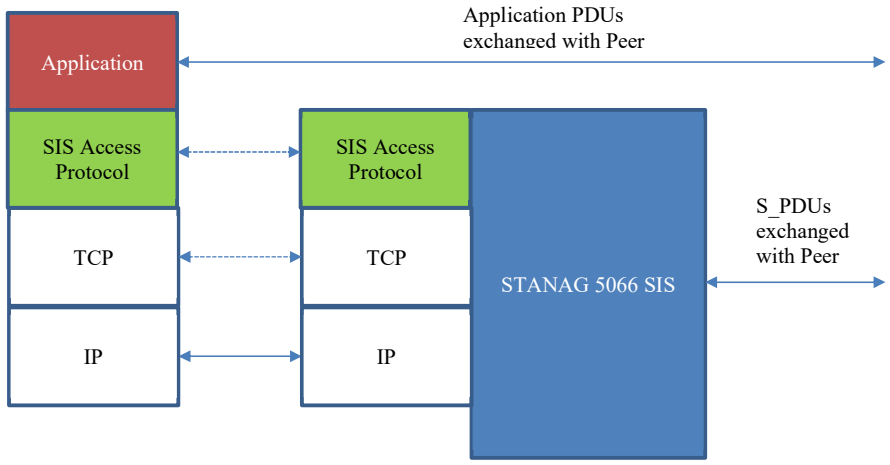


Figure S-2: SIS Access Protocol Mapping onto TCP

Figure S-2~~Figure S-2~~ shows how the SIS Access Protocol maps onto TCP/IP to connect an application to a SIS service. The SIS service communicates with its peer using S_PDUs. The application operating over SIS communicates with its peer using PDUs specific to the application. The SIS Access Protocol is a local protocol which does not have end to end interactions. It simply enables an application to interact with the local SIS service.

Annex A defines the core SIS services, which are extended by additional management and flow control services specified in this annex. Each of these service primitives is defined in a manner that is simple flow of data in one direction between application and SIS. This annex defines a framed encoding of each primitive, such that it can be mapped directly onto a TCP stream. This enables the application to access the SIS service over a TCP stream.

The Default SIS Port Number for the Raw SIS Socket Server **shall** be the decimal number '5066', a number registered with the Internet Assigned Number Authority (IANA) for this purpose. STANAG 5066 implementations, both clients and subnetworks, **should** be configurable to use any other valid port number as the default.

A SIS Access Protocol Server shall listen on the Default SIS Port Number for connection requests from clients. The Server shall accept sixteen connections simultaneously, so that every SAP can be served.

S_PRIMITIVES **shall** be sent over the SIS Access Protocol without any separating

characters or framing data other than the generic S_PRIMITIVE encoding elements defined in this annex. No other messages between client and subnetwork **shall** be sent. Implementation-dependent communication between a client and subnetwork over the Raw SIS-Socket-Server Interface for the purposes of system management **may** be encapsulated within the S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT _MSG_INDICATION primitives defined in this annex. [NB: This does not preclude implementation-dependent communication over another socket on a different port number, but such use is outside of the scope of this standard and is not recommended.]

S.3. Core Services

The primary goal of the SIS protocol is to support client access to the core SIS services, which are specified as the SIS service in Annex A.

S.4. Management & Flow Control Services

In addition to the core services, there are some additional services provided by the SIS protocol that pertain to connection management of the SIS protocol link.

Table A-1. Management and Flow Control Primitives

CLIENT -> SUBNETWORK INTERFACE	SUBNETWORK INTERFACE -> CLIENT
	S_SUBNET_AVAILABILITY (Subnet Status, Reason)
	S_DATA_FLOW_ON( )
	S_DATA_FLOW_OFF ( )
S_MANAGEMNT _MSG_REQUEST (MSG TYPE, MSG BODY)	S_MANAGEMENT_MSG_INDICATION (MSG TYPE, MSG BODY)
S_KEEP_ALIVE ( )	S_KEEP_ALIVE ( )

Table A-1 summarizes primitives to specify management and flow control operations, which extend the core SIS service primitives specified in Annex A. The details of each of these primitives is set out below.

S.4.1. Interface Flow Control Primitives: S_DATA_FLOW_ON and S_DATA_FLOW_OFF

Name :

S_DATA_FLOW_ON
S_DATA_FLOW_OFF

Arguments :
NONE
Direction :
Subnetwork Interface-> Client

Description :

The S_DATA_FLOW_ON and S_DATA_FLOW_OFF primitives **shall** (1) be issued by the Subnetwork Interface Sublayer to control the transfer of U_PDUs submitted by a client.

On receipt of an _DATA_FLOW_OFF primitive, the client **shall** (2) cease transferring U_PDUs over the interface.

Transfer over the interface of U_PDUs by the client **shall** (3) be enabled following receipt of an S_DATA_FLOW_ON primitive.

The Subnetwork Interface Sublayer can use these two primitives (or other mechanisms) to control the flow of data from locally attached clients. U_PDUs from an attached client to which the S_DATA_FLOW_OFF primitive has been sent may be discarded by the Subnetwork Interface Sublayer without acknowledgement, indication, or warning.

A client **shall** (4) not control the flow of data *from* the subnetwork by any mechanism, explicit or implicit.

All clients **shall** (5) be ready to accept at all times data received by the HF Node to which it is bound; clients not following this rule may be disconnected by the node.

S.4.2. S_MANAGEMENT_MSG_REQUEST Primitive

Name :

S_MANAGEMENT_MSG_REQUEST

Arguments :

1. MSG TYPE
2. MSG BODY

Direction :

Client-> Subnet Interface

Description :

The S_MANAGEMENT_MSG_REQUEST primitive **shall** (1) be issued by a client to submit a “Management” message to the Subnetwork.

The complex argument MSG may be implementation dependent and is not specified in this edition of the standard. At present, a minimally compliant HF subnetwork implementation **shall** (2) be capable of receiving this primitive, without further requirement to process its contents.

Depending on the value of the complex argument *MSG*, this primitive can take the form of a Command (e.g. Go-To-EMCON, Go-Off-Air, etc.) or of a Request (e.g. Request-For-Subnetwork-Statistics, Request-For-Connected-client-Information, etc.).

Note that this primitive is not intended to allow for the transmission of management coordination messages over the air. This is an interaction between peer subnet management clients and as such shall be accomplished using the UNIDATA primitive defined elsewhere in this annex.

S.4.3. S_MANAGEMENT_MSG_INDICATION Primitive

Name :

S_MANAGEMENT_MSG_INDICATION

Arguments :

1. MSG TYPE
2. MSG BODY

Direction :

Subnetwork Interface-> Client

Description :

The S_MANAGEMENT_MSG_INDICATION primitive **shall** (1) be issued by the Subnetwork to send a "Management" message to a client.

The complex argument MSG may be implementation dependent and is not specified in this edition of the standard. At present, a minimally compliant client **shall** (2) be capable of receiving this primitive, without further requirement to process its contents.

As implementation options, the complex argument *MSG* could take several values such as: Subnetwork- Statistics, Connected-client-Information, etc. This primitive could be issued either in response to a S_MANAGEMENT_MSG_REQUEST or asynchronously by the Subnetwork.

S.4.4. S_KEEP_ALIVE Primitive

Name :

S_KEEP_ALIVE

Arguments :

NONE

Direction :

Client-> Subnetwork
Interface Subnetwork
Interface-> Client

Description :

The S_KEEP_ALIVE primitive can be issued as required (e.g. during periods of inactivity) by the clients and/or the Subnetwork Interface to sense whether the physical connection between the client and the Subnetwork is alive or broken. This primitive may be redundant if the implementation of the physical connection provides an implicit mechanism for sensing the status of the connection.

A minimally compliant implementation of a client or subnetwork interface is not required to generate the S_KEEP_ALIVE primitive except in response to the receipt of an S_KEEP_ALIVE primitive.

When the S_KEEP_ALIVE Primitive is received, the recipient (i.e, client or Subnetwork Interface) **shall**
respond with the same primitive within 10 seconds.

If a reply is not sent within 10 seconds, no reply **shall** (2) be sent.

A client or Subnetwork Interface **shall** (3) not send the S_KEEP_ALIVE Primitive more frequently than once every 120 seconds to the same destination.

S.4.5. S_SUBNET_AVAILABILITY Primitive

Name :

S_SUBNET_AVAILABILITY

Arguments :

1. Node Status
2. Reason

Direction :

Subnetwork Interface-> Client

Description:

The S_SUBNET_AVAILABILITY primitive may be sent asynchronously to all or selected clients connected to the Subnetwork Interface Sublayer to inform them of changes in the status of the node to which they are attached. For example, clients can be informed using this primitive that available resources (e.g., bandwidth) have been temporarily reserved for other clients. Alternatively, this primitive could be used to inform clients that the node has entered an EMCON state and as a result they should only expect to receive Data and will not be allowed to transmit data.

The contents of this primitive are implementation dependent and not specified in this edition of the standard. At present, a minimally compliant client implementation **shall** (1) be capable of receiving this primitive, without further requirement to process its contents.

As implementation options, the *Node Status* argument could specify the new Status of the node. Possible values of this argument could be ON, OFF, Receive-Only, Transmit-Only-to-Specific-Destination- Node/SAP, etc.

Node Status	Value
OFF	0
ON	1
Receive-Only	2
Half-Duplex	3
Full-Duplex	4
Transmit-Only	5

If the Subnetwork Status is other than ON, the *Reason* argument explains why. Values of this argument shall be as specified below.

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

Reason	Value
unspecified	0
Local Node in EMCON	1
Reserved	2

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

S.5. Encoding of Primitives

The encoding of the S_Primitives for communication using the protocol specified in this annex **shall** (1) be in accordance with text and figures in the subsections below.

S.5.1. **Generic Field Encoding Requirements**

Unless noted otherwise, the bit representation for argument values in an S_Primitive **shall** (1) be encoded into their corresponding fields in accordance with CCITT V.42, 8.1.2.3, which states that:

- when a field is contained within a single octet (i.e, eight bit group), the lowest bit number of the field **shall**(2) represent the lowest-order (i.e., least-significant-bit) value;
- when a field spans more than one octet, the order of bit values within each octet **shall** (3) progressively decrease as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.

The 4-byte address field in the S_primitives **shall** (4) carry the 3.5-byte address and address-size information defined in Section Node ADDRESS Encoding for all Primitives~~Node ADDRESS Encoding for all Primitives~~S.5.16.1. The lowest order bit of the address shall be placed in the lowest order bit position of the field (generally bit 0 of the highest byte number of the field), consistent with the mapping specified in Section C.3.2.6 for D_PDUs.

Formatted: Font: (Default) Arial, 12 pt

S.5.2. **S_Primitive Generic Elements and Format**

As shown in Figure A-1(a), all primitives **shall** (1) be encoded as the following sequence of elements:

- a two-byte S_Primitive preamble field, whose value is specified by the 16-bit Maury-Styles sequence below;

- a one-byte version-number field;

- a two-byte Size_of_Primitive field;

- a multi-byte field that contains the encoded S_Primitive.

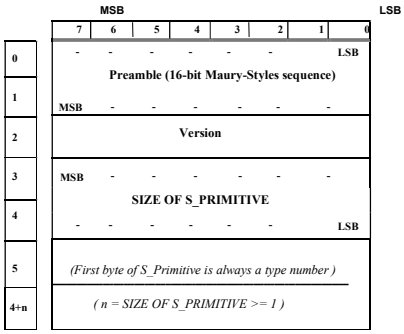


Figure S-3: Element-Sequence *Encoding of "S_" Primitives*

The S_Primitive preamble field **shall** (2) be encoded as the 16-bit Maury-Styles sequence shown below, with the least significant bit (LSB) transmitted first over the interface:

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

i.e., with the multi-byte S_Primitive field represented in hexadecimal form as 0xEB90, the least-significant bits of the sequence **shall** (3) be encoded in the first byte (i.e, byte number 0) of the preamble field and the most significant bits of the sequence **shall** (4) be encoded in the second byte (i.e, byte number 1) of the preamble field as follows:

	MSB 7	6	5	4	3	2	1	LSB 0	
0	1	0	0	1	0	0	0	0	0x90
1	1	1	1	0	1	0	1	1	0xEB

Figure S-4: Encoding of Maury-Styles Preamble-Sequence in "S_" Primitives

[***Note***: This encoding of the Maury-Styles preamble sequence is an exception to the general requirement of section S.5.1 for field encoding.]

Following the Maury-Styles sequence, the next 8 bit (1-byte) field **shall** (5) encode the 5066 version number. For this edition of the standard, the version number **shall** (6) be all zeros, i.e, the hexadecimal value 0x00, as shown in Figure S-3~~Figure S-3~~.

The next 16 bit (two-byte) field **shall** (7) encode the size in bytes of the S_primitive-dependent field to follow, exclusive of the Maury-Styles sequence, version field, and this size field. The LSB of the of the size value **shall** be mapped into the low order bit of the low-order byte of the field as shown in Figure S-3~~Figure S-3~~.

Unless specified otherwise, the order of bit transmission for each byte in the encoded S_Primitive **shall** (9) be as described in CCITT V.42 paragraph 8.1.2.2, which specifies the least significant bit (LSB, bit 0 in the figures below) of byte 0 **shall** (10) be transmitted first.

The sixth byte (i.e., byte number 5) of the sequence **shall** (11) be the first byte of the encoded primitive and **shall** be equal to the S_Primitive type number, with values encoded in accordance with the respective section that follows for each S_primitive

The remaining bytes, if any, in the S_Primitive **shall** (13) be transmitted sequentially, also beginning with the LSB of each byte, in accordance with the respective section that follows for each S_primitive.

In the subsections that follow, any bits in a S_Primitive that are specified as NOT USED **shall** (13) be encoded with the value “0” unless specified otherwise for the specific S_Primitive being defined.

S.5.3. S_BIND_REQUEST Encoding

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

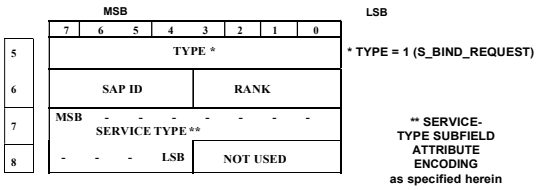
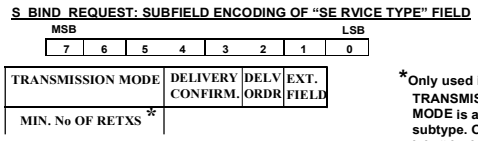


Figure S-5: Encoding of S_BIND_REQUEST Primitive

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



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

Figure S-6: Sub-field Attribute Encoding of S_BIND_REQUEST SERVICE-TYPE field.

Argument : SERVICE
TYPE Primitive :
S_BIND_REQUEST

The SERVICE TYPE argument **shall** (3) specify the default type of service requested by the client. This type of service **shall** (4) apply to any U_PDU submitted by the client until the client unbinds itself from the node, unless overridden by the DELIVERY MODE argument of the U_PDU. A client **shall** (5) change the default service type only by unbinding and binding again with a new S_BIND_REQUEST.

The RANK value was present in Edition 3 of STANAG 5066, but is not used in this edition. It should be set to zero on transmission and ignored on reception.

The SERVICE TYPE argument is complex, consisting of a number of attributes encoded as sub-fields. Although the exact number of attributes and their encoding is left for future definition and enhancement using the Extended Field attribute, the following attributes are mandatory:

1. *Transmission Mode for the Service.* --- ARQ or Non-ARQ Transmission Mode **shall**(6) be specified, with one of the Non-ARQ submodes if Non-ARQ was requested. A value of "0" for this attribute **shall**(7) be invalid for the SERVICE TYPE argument when binding. Non-ARQ transmission can have submodes such as: *Error-Free-Only* delivery to destination client, delivery to destination client even with *some* errors.

2. *Data Delivery Confirmation for the Service* --- The client **shall** (8) request one of the Data Delivery Confirmation modes for the service. There are three types of data delivery confirmation:
   - None
   - Node-to-Node Delivery Confirmation
   - Client-to-Client Delivery Confirmation

   The client can request explicit confirmation, i.e, Node-to-Node or Client-to-Client, from the Subnetwork to provide indication that its U_PDUs have been properly

delivered to their destination. Explicit delivery confirmation **shall** (9) be requested only in combination with ARQ delivery.

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

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

4. *Extended Field* --- Denotes if additional fields in the SERVICE TYPE argument are following; at present this capability of the SERVICE TYPE is undefined, and the value of the Extended Field Attribute **shall** (11) be set to "0".

5. *Minimum Number of Retransmissions* --- This argument **shall** (12) be valid if and only if the Transmission Mode is a Non-ARQ type. If the Transmission Mode is a Non-ARQ type, then the subnetwork **shall** (13) retransmit each U_PDU the number of times specified by this argument. This argument may be "0", in which case the U_PDU is sent only once.

   [Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]

S.5.4. S_UNBIND_REQUEST Encoding

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

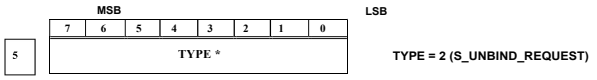


Figure S-7: Encoding of S_UNBIND_REQUEST Primitive

S.5.5. **S_BIND_ACCEPTED Encoding**

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

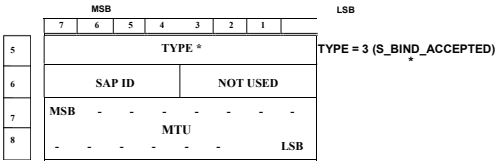


Figure S-8: Encoding of S_BIND_ACCEPTED Primitive

S.5.6. **S_BIND_REJECTED Encoding**

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

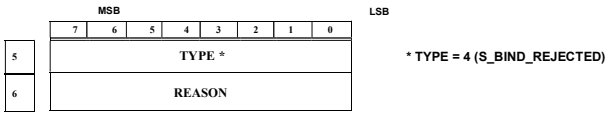


Figure S-9: Encoding of S_BIND_REJECTED Primitive

The Reason **shall** be set to a value specified in A.2.2.4 of Annex A.

S.5.7. **S_UNBIND_INDICATION Encoding**

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

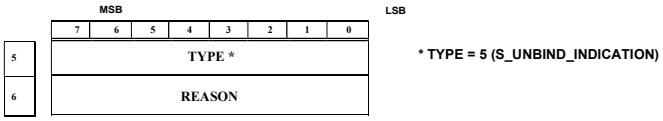


Figure S-10: Encoding of S_UNBIND_INDICATION Primitives

The Reason **shall** be set to a value specified in A.2.2.5 of Annex A.

S.5.8. **S_SUBNET_AVAILABILITY Encoding**

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

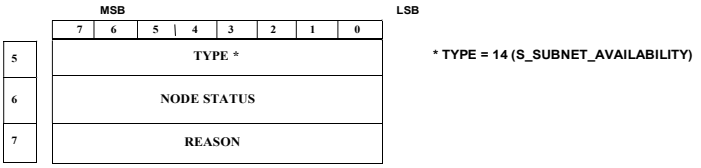


Figure S-11: Encoding of S_SUBNET_AVAILABILITY Primitives.

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

S.5.9. **S_DATA_FLOW_ON and S_DATA_FLOW_OFF Encoding**

The S_DATA_FLOW_ON and S_DATA_FLOW_OFF primitives **shall** (1) be encoded as one-byte fields as follows:

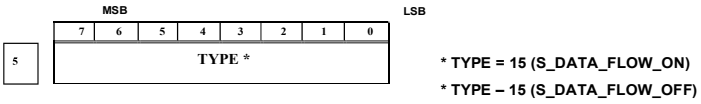


Figure S-12: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

S.5.10. **S_KEEP_ALIVE Encoding**

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

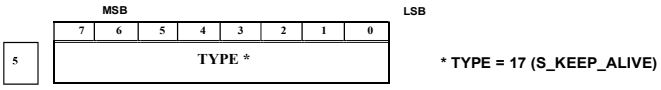


Figure S-13: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

S.5.11. S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_ MSG_INDICATION Encoding

The S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_ INDICATION primitives **shall** (1) be encoded as implementation-dependent variable-length fields as follows:

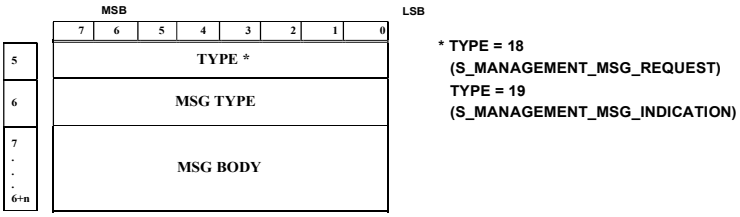


Figure S-14: Encoding of S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_INDICATION Primitives.

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

S.5.12. S_UNIDATA_REQUEST Encoding

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

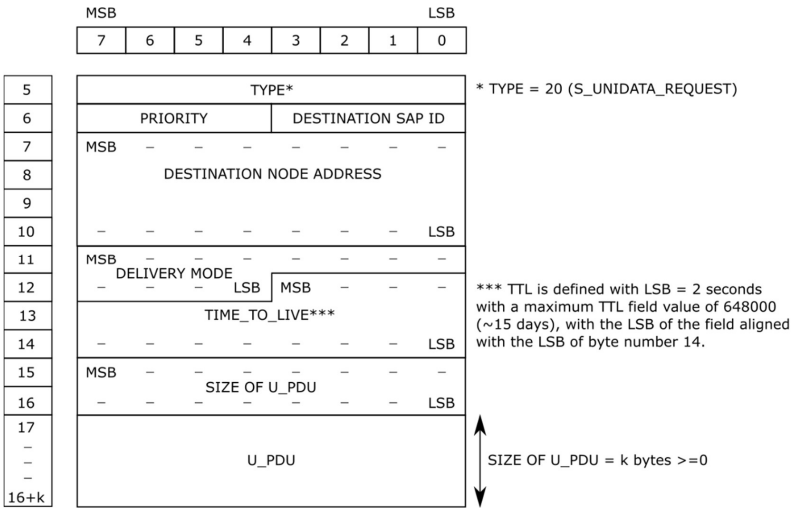


Figure S-15: Encoding of S_UNIDATA_REQUEST Primitives.

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in SectionS.5.16.1.

The DELIVERY MODE field **shall** (3) be encoded as specified in Section S.5.16.2.

S.5.13. **S_UNIDATA_INDICATION Encoding**

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

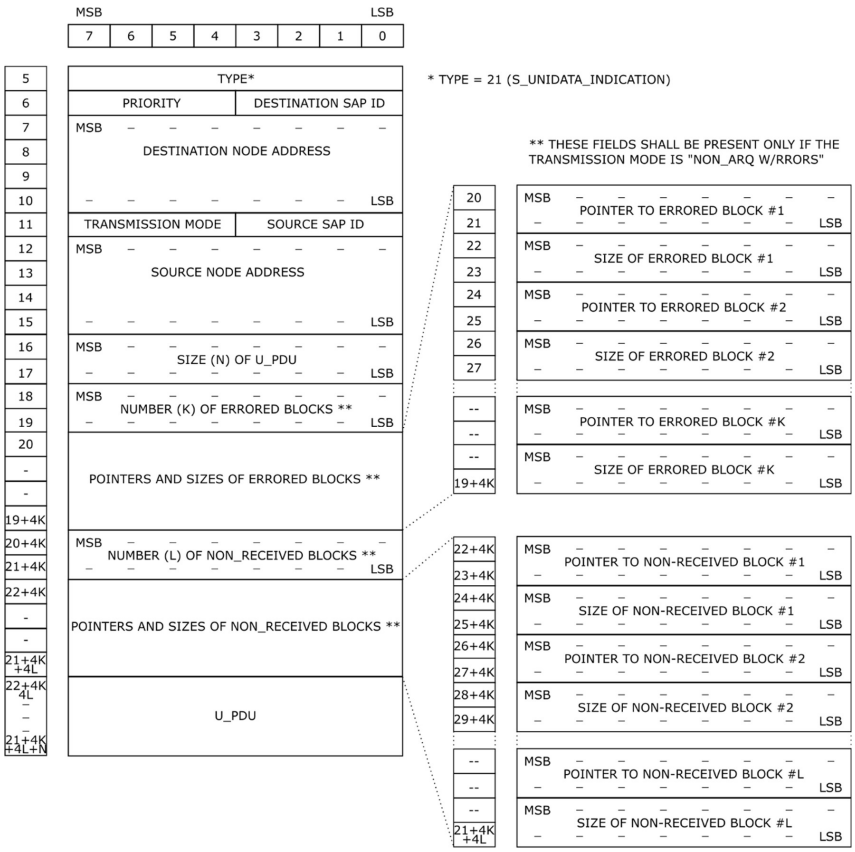


Figure S-16: Encoding of S_UNIDATA_INDICATION Primitives

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** (2) be encoded as specified in Section S.5.16.1.

The TRANSMISSION MODE field **shall** (3) be encoded as specified in Section S.5.16.3.

S.5.14. **S_UNIDATA_REQUEST_CONFIRM Encoding**

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

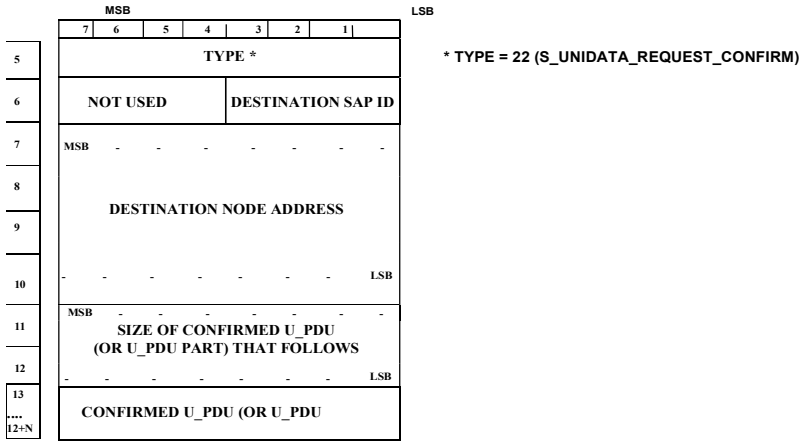


Figure S-17: Encoding of S_UNIDATA_REQUEST_CONFIRM Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section S.5.16.1.

S.5.15. **S_UNIDATA_REQUEST_REJECTED Encoding**

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

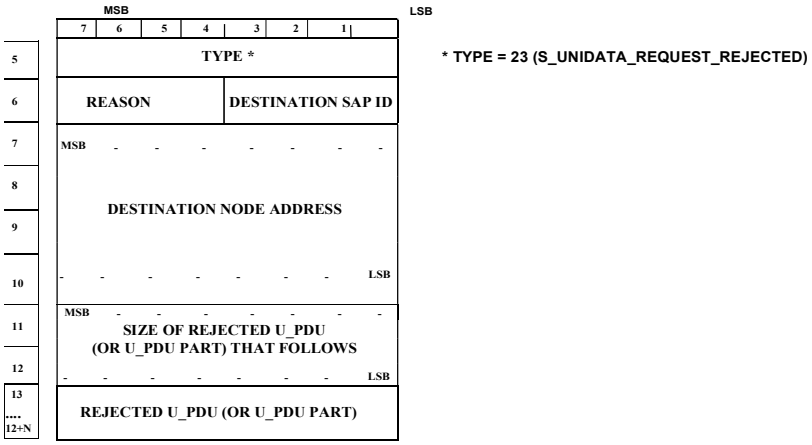


Figure S-18: Encoding of S_UNIDATA_REQUEST_REJECTED Primitives.

The DESTINATION NODE ADDRESS field **shall** (2) be encoded as specified in Section S.5.16.1.

The Reason **shall** be set to a value specified in A.2.2.8 of Annex A.

S.5.16. **Additional S_Primitive Encoding Requirements: Encoding of Common Fields**

In order to clarify some of the procedures and tasks executed by the sublayers, additional details concerning some of the arguments of the Primitives described in previous sections are provided below.

S.5.16.1. Node ADDRESS Encoding for all Primitives

Arguments : SOURCE NODE ADDRESS, DESTINATION NODE ADDRESS, or REMOTE NODE ADDRESS
Primitives : ALL “UNIDATA” primitives.

For reduced overhead in transmission, node addresses **shall** (1) be encoded in one of several

formats that are multiples of 4-bits (“half-bytes”) in length, as specified in Figure S-19~~Figure S-19~~.

All node addresses are logically 7 half bytes, The compact encoding here reflects the encoding in Annex C, where leading half bytes of value zero are omitted. Leading zero half bytes of value zero **may** be omitted to enable encoding of the address in a smaller number of half bytes. Leading zeros may **also** be encoded in an address using all seven half bytes.

Addresses that are encoded as Group node addresses **shall** (2) only be specified as the Destination Node address of Non-ARQ PDUs.

Destination SAP IDs and destination node addresses of ARQ PDUs and source SAP IDs and source node addresses of all PDUs **shall** (3) be individual SAP IDs and individual node addresses respectively.

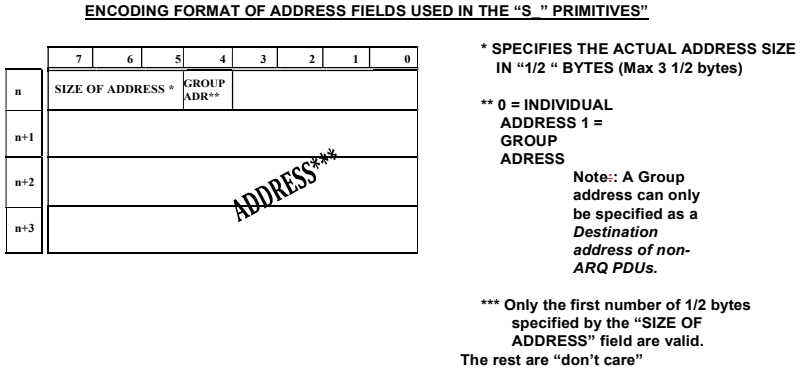


Figure S-19: Encoding of Address Fields in S_Primitives.

S.5.16.2. Delivery-Mode Encoding for the S_UNIDATA_REQUEST Primitive

Argument : DELIVERY MODE
Primitive : S_UNIDATA_REQUEST

The DELIVERY MODE is a complex argument consisting of a number of attributes, as specified here. The DELIVERY MODE argument **shall** (1) be encoded as shown in Figure S-20~~Figure S-20~~.

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

The attributes of this argument are similar to those described in the *Service Type* argument of the S_BIND_REQUEST:

Formatted: Font: 10.5 pt

Formatted: Font: Bold

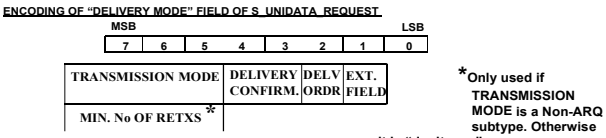
Formatted: Font: Bold

Formatted: Font: 10.5 pt

1. *Transmission Mode of this U_PDU.* --- ARQ or Non-ARQ Transmission can be requested. A value of "0" for this attribute **shall** (4) equal the value "DEFAULT" for the Delivery Mode. If the DELIVERY MODE is "DEFAULT", all other attributes encoded in the argument **shall** (5) be ignored.
2. *Data Delivery Confirmation for this PDU* --- None, node-to-node, or client-to-client.

3. *Order of delivery of this PDU to the receiving client.* --- A client may request that its U_PDUs are delivered to the destination client "in-order" (as they are submitted) or in the order they are received by the destination node.
4. *Extended Field* --- Denotes if additional fields in the DELIVERY MODE argument are following; at present this capability of the DELIVERY MODE is undefined, and the value of the Extended Field Attribute **shall** (6) be set to "0".
5. *Minimum Number of Retransmissions* --- This argument **shall** (7) be valid if and only if the Transmission Mode is a Non-ARQ type or sub-type. If the Transmission Mode is a Non-ARQ type or subtype, then the subnetwork **shall** (8) retransmit each U_PDU the number of times specified by this argument. This argument may be "0", in which case the U_PDU is sent only once.

   [Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]

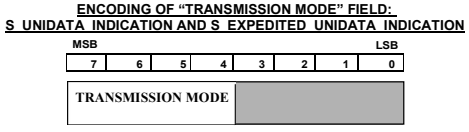


TRANSMISSION MODE	DELIVERY CONFIRMATION	DELIVERY ORDER	EXTENDED FIELD
0000 = Ignore Service type field. 0001 = ARQ 0010 = Non-ARQ (e.g., Broadcast) 0011 = Non-ARQ WITH ERRORS	00 = No Confirmation 01 = Node Delivery Conf. 10 = Client Delivery Conf. 11 = Not defined	0 = In-order Delivery 1 = As they arrive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

Figure S-20: Encoding of the Delivery Mode field in the S_UNIDATA_REQUEST and primitives

S.5.16.3. TRANSMISSION-MODE Encoding for the S_UNIDATA_INDICATION Primitive

Argument: TRANSMISSION-MODE
S_Primitives: S_UNIDATA_INDICATION



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

Figure S-21: Encoding of Transmission Mode Field in S_UNIDATA_INDICATION primitive.

The subnetwork notifies a client of the transmission-mode used to deliver a U_PDU Argument with the TRANSMISSION-MODE argument. The TRANSMISSION-MODE argument in the S_UNIDATA_INDICATION Primitives **shall** (1) be encoded as shown in Figure S-21~~Figure S-21~~.

[Note: The unused bits in this argument are allocated to the SOURCE SAP_ID argument encoding for the S_UNIDATA_INDICATION Primitive.]

Formatted: Font: 10.5 pt

S.6. Deprecated Services

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

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

S.6.1. Hard Links

This optional deprecated service is defined as the HARD LINKS profile option. The service is specified in Annex A.

The encoding of Hard Links protocol Elements in the SIS Access Protocol is specified in Edition 3 Annex A, Section A.3. Use of Rank may be necessary as part of the Hard Links service.

S.6.2. Expedited Data

This optional deprecated service is defined as the EXPEDITED DATA profile option. The service is specified in Annex A.

The encoding of Expedited Data protocol Elements in the SIS Access Protocol is specified in Edition 3 Annex A, Section A.3.

S.7. Changes in Edition 4

This Annex is built from information in Edition 3 Annexes A and F. It supports the

services defined in Annex A, and the protocol specified is unchanged from Edition 3.

It does not specify protocol for the hard link and expedited services, which are defined only in Edition 3.