Annex F. Subnetwork Client Definitions (information only)

This Annex defines interactions with the subnetwork for a set of clients types (i.e., subnetwork clients) to which the HF subnet defined in the mandatory part of this STANAG provides data transport service. These clients types are defined in order to provide an initial set of interoperable applications that make use of the HF subnet. Note that with respect to the higher-layer peer-to-peer interaction between a pair of subnetwork clients, one subnetwork client may act as a higher-layer server to another subnetwork client (e.g., for a mail transfer service in which one subnetwork client is a User Agent and another subnetwork client is the Message Transfer Agent).

Each section of the Annex describes the actions of two different subnetwork clients that carry on a client-server relationship between themselves over the subnetwork. This annex is a minimum requirement; compliance with this annex does not prohibit additional interactions that are not defined. Table F-1 defines standard SAP IDs for the client applications defined in this annex.

Table F-1: SAP ID Assignments

Client Application Type	SAP ID	Defined in Annex
Subnet management client	0	F.1
Acknowledged message	1	F.2
Un-acknowledged message	2	F.3
HMTP	3	F.4
HFPOP	4	F.5
Operator orderwire	5	F.6
Reliable Connection-Oriented Protocol	6	F.7
Unreliable Datagram Oriented Protocol	7	F.8
PPP client	8	F.9
IP client	9	F.10
reserved for future assignment	10-11	
available for arbitrary use	12-15	

F.1 Subnet Management Client/Server

This section defines minimal requirements for control of the local node using the S_MANAGEMENT_MSG_REQUEST, S_MANAGEMENT_MSG_INDICATION, and S_SUBNET_AVAILABILITY primitives, and for coordination with distant subnet management clients or agents using the S_UNIDATA primitives.

Subnetwork management clients shall attach to the subnetwork using SAP ID 0. The rank of the client must be 15 if the client is entitled to submit commands that will change the configuration of the node or subnetwork.

The nature and specification of peer-to-peer level communication between Subnetwork Management Clients is beyond the scope of this STANAG. It is strongly recommended that existing standards be used for network management such as the Simple Network Management Protocol (SNMP) or Common Management Information Protocol (CMIP).

F.2 Acknowledged-Message Client/Server

This section defines an application-layer client-server protocol that supports point-to-point Acknowledged Message transfer over a radio link using the HF subnet described in the mandatory part of this STANAG. This would be suitable, for example, to replace the current text-based ship-shore systems used in NATO. The client-server relationship is defined with respect to the sender of the message (i.e., the client) and the receiver of the message (i.e., the server). Applications that support bi-directional message transfer between nodes would implement the Acknowledged-Message protocol to support both the client and the server functions.

F.2.1 Service Model Overview: Sending vs. Mailing

The primary difference between the sending service defined in this subsection and the mail service defined in Annex F.4 for the HMTP application is that the Acknowledged-Message-Service is not required to support mail forwarding. Sending supports transfer of text-based messages from a local user at one node to a local user at another node, without message forwarding at the application level.

The Acknowledged-Message Service is based on the Basic SMTP Protocol as defined in Internet Standard 10, RFC 821, with modified Command-Pipelining as defined in Internet RFC. The Acknowledged-Message Service is a text-based message transfer service and does not support the SMTP Service Extensions for multi-part message bodies and Multimedia Internet Mail Extensions (i.e., MIME and support for binary attachments to text-based messages). In all other respects the Acknowledged-Message Service mimics the HF Mail Transfer Application of Annex F.4 (which in turn mimics the Internet Simple-Mail Transfer Protocol [SMTP] Model).

F.2.2 Acknowledged Message Service: Command Set

The following commands, taken as a minimal subset from the SMTP command set, are defined to support the Acknowledged-Message Application. These are used in the client/server transaction for the Acknowledged Message Service instead of the SMTP MAIL commands defined in Annex F.4 (which allows forwarding).

Commands are sent from the Acknowledged-Message Client to the server via the HF subnetwork. Commands are text-based strings encoded in the ASCII code, terminated by a carriage-return/linefeed character, one code-character bit aligned in each of the octet in a user protocol data unit (U_PDU) carried by the subnetwork.

F.2.2.1 EHLO ---- HELLO w/ Extensions

This command is used in the same manner for the Acknowledged-Message Service as for the HMTP service defined in Annex F.4; however, the range of acceptable extensions to the basic SMTP service is limited to the modified Command-Pipelining Extensions defined herein. In particular other SMTP extensions for MIME or binary attachments are not supported.

The EHLO command establishes the connection between the Acknowledged-Message Client and the message-server at the destination. The argument field may contain the host name of the Acknowledged-Message client, as follows:

EHLO <SP> <*sending_host_name*> <CRLF>

The response to this command shall be limited to an OK reply (Reply Code 250) acknowledging the <*sending_host_name*>, and an OK reply indicator that command pipelining is supported.

Multiple messages may be sent from the client to the server following the EHLO command. Each message must be encapsulated by the SEND, RCPT, DATA, and data-termination commands defined below.

F.2.2.2 SEND

The SEND command is used to initiate a transaction in which the Acknowledged-Message data is delivered to one or more local message stores at the receiver. The argument field contains a reverse-path designating the sender's message store, as follows:

SEND <SP> FROM:<*reverse-path*> <CRLF>

The <*reverse-path*> argument is of form <*sending_message_store*>, which is the name of the mailbox on the client. Since message forwarding to another node or host is not supported by the Acknowledged Message service, the sending host or node name is not required.

The SEND command is successful if the message is delivered to the specified message store at the destination. A reply code 250 shall be sent as indication of the successful delivery of the message.

The SEND command requires that the message data be delivered to the destination user's local message store, if it exists. If the user does not have a local message store with the SEND server, reply code 450 reply may be returned to a RCPT command.

Other appropriate reply codes from the same set that is used for the HMTP Mail commands of Annex F.4 may be used in reply to the commands.

The SEND command clears the receiving server's reverse-path buffer, the forward-path buffer, and the mail data buffer, and then inserts the reverse-path information from this command into the reverse-path buffer.

Multiple messages are sent by multiple occurrences of the SEND command, each as the initial command for a valid message transfer sequence consisting of the SEND command, one or more RCPT commands, the DATA command, the message data, and the data-termination command. Occurrence of a SEND command is valid only following an EHLO command or the data-termination command.

F.2.2.3 RCPT (RECIPIENT)

The RCPT command is used to identify an individual recipient of the mail data; multiple recipients for the same message are specified by multiple instances of the RCPT command following a SEND command. The RCPT command is used in the same manner for the Acknowledged-Message Service as for the HMTP service defined in Annex F.4, and further specification on its use is provided there. Since message forwarding to another node or host is not supported by the Acknowledged Message service, the sending host or node name is not required. Arguments to the RCPT command may therefore be a full path name as in Annex F.4 (i.e, <*message_store@destination_node*>, or a short path name, where the destination node is implied, as follows:

RCPT <SP> TO:<*message_store@destination_node*> <CRLF>

or, equivalently,

RCPT <SP> TO:<*message_store* > <CRLF>

F.2.2.4 DATA

The DATA command marks the beginning of the message data. The Acknowledged Message Server treats the lines following the DATA command as message data from the sender. This command causes the message data from this command to be appended to the message data buffer. This command is used in the same manner for the Acknowledged-Message Service as for the HMTP service defined in Annex F.4, and further specification on its use is provided there.

In the Acknowledged Message Service, lines following the DATA Command are restricted to be any of the 128 lower-order ASCII character codes (i.e, codes whose most-significant bit equals zero).

The message data is terminated in the same manner as defined for the HMTP Service in Annex F.4, that is, by the character sequence "<CRLF>.<CRLF>". Applications implementing the Acknowledged Message Service shall implement the Data Transparency procedure defined in Annex F.4 to ensure that the recipient-server does not confuse the message data itself with a valid message-terminator.

F.2.2.5 RSET (RESET)

A client uses the RSET command to abort the current message transaction. Any stored sender, recipients, and message data must be discarded by the receiving server, and all buffers and state tables cleared. The receiving server then must send an OK reply. This command is used for the Acknowledged-Message Service in the same manner as for the HMTP service defined in Annex F.4.

F.2.2.6 QUIT

A client uses the QUIT command to end the mail transfer session, and to begin closing the connection. This command specifies that the receiving server must send an OK reply, and then

close the reliable connection between the client and server. This command is used in the same manner for the Acknowledged-Message Service as for the HMTP service defined in Annex F.4.

The receiver should not close the transmission channel until it receives and replies to a QUIT command (even if there was an error). The sender should not close the transmission channel until it sends a QUIT command and receives the reply (even if there was an error response to a previous command). If the connection is closed prematurely the receiver should act as if a RSET command had been received (canceling any pending transaction, but not undoing any previously completed transaction), the sender should act as if the command or transaction in progress had received a temporary error (4xx).

F.2.3 Acknowledged Message Service: Reply Codes

Command replies are encoded using the same set of Reply Codes defined in Annex F.4. There are a limited number of replies that are sensible for this application.

F.2.4 Acknowledged Message Service: Sample Dialogs

Dialogs between the Acknowledged-Message Client and Server use command pipelining in a manner similar to that defined for SMTP service extensions (RFC-1869), with the exception that command pipelining is required rather than optional. As defined in the Internet RFC, there is an initial negotiation stage between client and server that determines if the service extensions are supported, allowing backward compatibility with servers that do not. In the Acknowledged-Message service defined here, command pipelining is an enforced requirement that requires no initial negotiation phase.

A nominal command sequence consists of the following steps:

- connection establishment between client and server, using the EHLO command;
- delivery of one or more messages encapsulated by source and destination indicators, each message consisting of a sequence of the following form:
    - a SEND command,
    - (one or more) RCPT commands,
    - a DATA command,
    - the message data,
    - a message-termination code;
- connection termination

Each reply code returned by the Acknowledged-Message server must match, in order, one of the commands in the client's sequence. The mapping may be many to one, i.e., their may be more than one reply code that matches a command, as for example, the EHLO command, which is matched by an OK reply code for the sending node name and for the use of pipelining.

Note that, while commands from the Acknowledged-Message client must be grouped together without waiting for the corresponding replies from the server, there is no requirement that the server must receive all the client's commands in the sequence before it can initiate replies. For

example, the command sequence might be contained in more than one of the HF subnetwork's user protocol data units. The application-level client-server dialog allows the subnetwork to deliver in order without waiting for the entire command sequence to be received (indeed, the subnetwork would have no real indication of what is contained in the command sequence to know when an entire command group had been received). Command-pipelining for the HF subnetwork has two aspects:

1) commands are grouped and sent by the client without waiting for server replies, and
2) the server can initiate replies for those commands it has received without waiting to receive the remaining commands in the group.

An example of an application level dialog for the Acknowledged-Message Service is given in the subsection below.

F.2.4.1 Single Message, Three Recipients, Successful Delivery to Two Recipients

In this example, a connection is opened between the Acknowledged-Message client and server for the transmission of one message to message stores for three different users. Delivery to two of the three destinations is successful, as denoted by the matching reply codes. Delivery to the third destination is unsuccessful.

The client sends the following sequence of commands:

C: EHLO <sending_node>
C: SEND FROM:<user1>
C: RCPT TO:<user2>
C: RCPT TO:<user3>
C: RCPT TO:<user4>
C: DATA
C: Blah blah blah...
C: ...etc. etc. etc.
C: <CRLF>.<CRLF>
C: QUIT

The reply sequence from the server matches up with each of the clients commands as follows (note that the EHLO command has two responses, one acknowledging the connection from the source node, and the second acknowledging the use of command pipelining):

S: 250 Source <sending_node> OK
S: PIPELINING OK
S: 250 Source <user1> OK
S: 250 Destination <user2.> OK
S: 550 Destination <user3> not known
S: 250 Destination <user4> OK
S: 250 DATA OK
S: 250 QUIT OK

The message has now been accepted for user2 and user4.

F.2.5 Acknowledged Message Service: Subnetwork Service Requirements

Clients and Servers for the Acknowledged Message Service bind to the HF Subnetwork at SAP ID 1. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

The commands, replies, and message data associated with the Acknowledged-Message service are submitted to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

The address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the host on which the destination message-client or server is located.

The encoded data for the commands, replies, and user message data for the Acknowledged-Message Service are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) or each character aligned with the LSB of the octet. Message data encoded as seven-bit quantities (as in earlier messaging systems) shall be bit-aligned, LSB to LSB, with the octets of the S_Primitive, and the eighth (i.e, MSB) of the octet set to zero.

F.3 Unacknowledged-Message Client/Server

This subsection defines an application-layer client-server protocol that supports unacknowledged transfer of messages over a radio link using the HF subnet described in the mandatory part of this STANAG. This would be suitable, for example, to replace the current text-based shore-ship broadcast systems. Because this type of service is inherently unreliable, mail services are not supported. It is assumed that any acknowledgement (positive or negative) for the receipt of messages sent using this application is the responsibility of the application level and communication for it is via some other channel.

For this service, there is the presumption that a one-way subnet connection exists between the message client and message-server; commands exist solely for the purpose of message delineation and routing from the source message store to the destination message store(s). The client simply sends messages which the receiving message server(s) will process and place in the designated message store(s) at the destination.

The encoded data for the commands and user message data for the Unacknowledged-Message Service are placed in the U_PDU in the same manner as defined in Annex F.2 for the Acknowledged Message Service.

For the Unacknowledged-Message Service, messages for broadcast transmission (i.e., transmission with non-ARQ service) are submitted using the normal S_UNIDATA_REQUEST primitives with the service-type set to non-ARQ. The address in the primitive will be either an individual address or a group address corresponding to the HF subnetwork address of the hosts on which the destination message stores are located.

Clients and Servers for the Unacknowledged Message Service bind to the HF Subnetwork at SAP ID 2. Client rank and priority are implementation-dependent parameters for this application. Use of Rank = 15 is discouraged. The default service requirements defined as follows:

- Transmission Mode = non-ARQ or non-ARQ w/ Errors
- Delivery Confirmation = NONE
- Deliver in Order = IN-ORDER DELIVERY

F.4 SMTP FOR USE OVER STANAG 5066 TRANSPORT (HMTP)

This annex defines an application-level client-server protocol which support acknowledged transfer of SMTP email messages over a radio link using the HF subnet described in the mandatory part of this STANAG. This would be suitable, for example, to provide the HF communications service for an application-level gateway of email over HF radio, using the inherent mail-forwarding capabilities defined for the SMTP. Alternatively, a Message Transfer (MTS) Service could use the HMTP and HF-subnetwork as the connection between a User Agent (UA) and Message Transfer Agent (MTA)

The presumption and requirement for an HMTP client and server is that they use the HF subnetwork protocol stack directly, without intervening transport or network protocol. SMTP mail clients/servers that incorporate TCP/IP protocols in their connection may use the HF subnetwork, but they interface to the subnetwork as an IP/PPP client rather than as an HMTP client as described here.

F.4.1. Introduction and Service Characteristics

The objective of HF Mail Transfer Protocol (HMTP) is to transfer mail reliably and efficiently over an HF radio link using the subnetwork protocols described in the mandatory part of this STANAG. The service characteristics for the HMTP protocol are as follows:

- Basic Simple Mail Transfer Protocol, in accordance with Internet Standard 10 and 11 (i.e., RFC 821[1] and 822), with service extensions as defined and modified herein;
- Service Extensions Mechanisms for SMTP, as defined in RFC 1869 and modified herein;
- Command-Pipelining Service Extensions for SMTP, as defined in RFC 2197 and further modified herein;
- MIME Extensions for SMTP, as defined in RFCs 2045, 2046, 2047, and 2049.

The scope of modifications made to these Internet standards-track specifications are limited to performance and efficiency issues when the specifications are used in conjunction with the HF subnetwork.

F.4.2. THE SMTP MODEL AND EXTENSIONS

The HMTP is a variant implementation of the Simple Mail Transfer Protocol SMTP (i.e., as specified by Internet Standard 10, RFC 821) with Service Extensions (i.e, as specified by RFC

[1] Postel, J., "Simple Mail Transfer Protocol", Information Sciences Institute, University of Southern California, Network Working Group Request for Comments, RFC 821, August 1982

1869[2]) that allow Command Pipelining (i.e., as specified by RFC 2197[3]). A discussion of the SMTP model and various service extensions is presented here.

The basic SMTP design is based on the following model of communication: as the result of a user mail request, the SMTP client establishes a two-way transmission channel to an SMTP server. The SMTP server may be either the ultimate destination or an intermediate. SMTP commands are generated by the SMTP client and sent to the SMTP server. SMTP replies are sent from the SMTP server to the SMTP client in response to the commands. A typical SMTP transaction is shown in the SMTP/Basic-Service column of Table F.2. Each command or response in the dialog is a line of text encoded in the ASCII format. The command/response dialog requires fourteen steps to establish the connection, designate the source and destination addresses for the mail, validate the source and destination addresses, transfer the mail message, acknowledge receipt of the mail, and close the connection.

Once the transmission channel is established by the HELO / 250<>OK dialog, the SMTP client sends a MAIL command indicating the sender of the mail. If the SMTP-server can accept mail it responds with an OK reply. The SMTP client then sends a RCPT command identifying a recipient of the mail. If the SMTP-server can accept mail for that recipient it responds with an OK reply; if not, it responds with a reply rejecting that recipient (but not the whole mail transaction). The SMTP client and SMTP-server may negotiate several recipients. When the recipients have been negotiated the SMTP client sends the mail data, terminating with a special sequence. If the SMTP-server successfully processes the mail data it responds with an OK reply. The SMTP dialog is purposely lock-step, one-at-a-time.

The SMTP Service Extensions for Pipelining group the client-server messages in a way that significantly reduces the number of transactions, while providing the same assurance of reliability in the protocol, albeit delayed until the server completes each response to a grouped set of client requests. A set of transactions for SMTP with Command Pipelining also is presented in Table F.2 , identical in function and result to that for the SMTP/Basic Service, but with fewer transactions required.

In order to increase efficiency and reduce response time over low data rate channels, the HMTP combines even more of the steps that are separate in the SMTP. Unlike SMTP w/Command-Pipelining which first checks for a valid response that confirms a peers capability to use the pipelined commands, HMTP proceeds under the assumption that the peer-level process is fully compliant with its pipelined/grouped SMTP commands. This streamlines the process to use the minimum number of transactions between the client and server, as shown in the final column of Table F.2 comparing the three approaches. The disadvantage is that if the peer-level mail process is not compliant with HMTP, then the transactions are lengthy to no purpose, since the mail will not be transferred correctly but the transmissions could take significant time on the channel before this is determined.

While not shown in the sample dialogs of Table F.2, the HMTP service shall support the Multimedia Internet Mail Extensions (MIME) for SMTP as specified in RFC 2045, RFC 2046, RFC 2047, and RFC 2049. These extensions allow the HMTP protocol to support mail delivery of binary attachments as defined in the respective Internet standards.

[2] Klensin, J. et al, "SMTP Service Extensions", Network Working Group Request for Comments, RFC 1869, November 1995

[3] Freed, N., "SMTP Service Extension for Command Pipelining", Network Working Group Request for Comments, RFC 2197, September 1997

Table F-2 - Comparison of Basic SMTP, SMTP w/ Pipelining (per RFC-2197) and HMTP

Transaction Number: Source (C/S)	SMTP (basic service, w/o Service Extensions)	SMTP w/ Command Pipelining	HMTP
1:C	HELO <server_name>	EHLO <server_name>	EHLO <server_name> MAIL FROM: <user@client_name> RCPT TO: <other@destination> RCPT TO: <another@destination> DATA Message blah, blah, blah More blah, blah, blah . QUIT
2:S	250 <client_name>	250 <client_name> 250 PIPELINING 250 8-BIT MIME	250 <client_name> 250 PIPELINING 250 sender <user@client_name> OK 250 recipient <other@destination> OK 250 recipient <another@destination> OK 354 enter mail, end with line containing only "." 250 message sent 221 goodbye
3:C	MAIL FROM: <user@client_name>	MAIL FROM: <user@client_name> RCPT TO: <other@destination> RCPT TO: <another@destination> DATA Message blah, blah, blah More blah, blah, blah .	
4:S	250 sender <user@client_name> OK	250 sender <user@client_name> OK 250 recipient <other@destination> OK 250 recipient <another@destination> OK 354 enter mail, end with line containing only "."	
5:C	RCPT TO: <other@destination>	QUIT	
6:S	250 OK	250 message sent 221 goodbye	
7:C	RCPT TO: <another@destination>		
8:S	250 OK		
9:C	DATA		
10:S	354 <enter full mail text, ending with a line that contains only a ".">		
11:C	Message blah, blah, blah More blah, blah, blah .		
12:S	250 message sent		
13:C	QUIT		
14:S	221 goodbye		

HMTP combines the MAIL command, RCPT command, and the mail data for multiple mail messages into a single transmission to reduce the delays associated with link turnaround associated with the dialog between HMTP-client and HMTP server. SMTP service extensions for Command-Pipelining perform a similar grouping of SMTP commands, but to remain backwards-compatible with SMTP servers that do not implement Command Pipelining, the service extensions do not start grouping commands until an initial handshake is completed between SMTP client and server. The HMTP protocol groups all commands to provide the greatest efficiency, but consequently is not interoperable with SMTP servers.

The HMTP server shall provide a relay capability for the client in accordance with the basic SMTP specification. This relay capability may be used to provide an application-level gateway capability between the HF subnetwork and other networks, for example, those with lower latency and higher throughput for which the other protocols are more suited. With respect to relay and routing, the argument to the MAIL command is in the form “user@hostname”, which specifies who the mail is from. The argument to the RCPT command is in the same form and specifies the ultimate destination of the mail. The HMTP server shall forward mail in accordance with the basic SMTP specification. A destination will be rejected only if the server can not understand it. Source routing shall not be used, as the HMTP model requires the server to have mail-routing information.

Details of the HMTP service are defined in the paragraphs below.

F.4.3 THE HMTP SPECIFICATIONS

Specification of the HMTP commands, replies and syntax follows, based heavily on material from RFCs 821, 822, and 2197, which provide large portions of the text. The HMTP commands are discussed immediately below. The HMTP replies are discussed in Section F.4.3.2.

F.4.3.1 HMTP COMMANDS

The HMTP commands define the mail transfer or the mail system function requested by the user. HMTP commands are character strings terminated by <CRLF>. The command codes themselves are alphabetic characters terminated by <SP> if parameters follow and <CRLF> otherwise. The syntax of mailbox names must conform to server site conventions.

The commands consist of a command code followed by an argument field. Command codes are four alphabetic characters. Upper and lower case alphabetic characters are to be treated identically. Thus, any of the following may represent the mail command:

MAIL Mail mail MaIl mAIl

This also applies to any symbols representing parameter values, such as "TO" or "to" for the forward-path. Command codes and the argument fields are separated by one or more spaces. However, within the reverse-path and destination arguments case is important. In particular, in some hosts the user "smith" is different from the user "Smith".

The argument field consists of a variable length character string ending with the character sequence <CRLF>. The receiver is to take no action until this sequence is received.

Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

The following are the basic HMTP commands taken from RFC 821:

EHLO <SP> <*domain*> <CRLF>

MAIL <SP> FROM:<*reverse-path*> <CRLF>

RCPT <SP> TO:<*destination*> <CRLF>

DATA <CRLF>

RSET <CRLF>

QUIT <CRLF>

A mail transaction involves several data objects that are communicated as arguments to different commands. The reverse-path is the argument of the MAIL command, the forward-path is the argument of the RCPT command, and the mail data is the argument of theDATA command. These arguments or data objects must be transmitted and held pending the confirmation communicated by the end of mail data indication that finalizes the transaction. The model for this is that distinct buffers are provided to hold the types of data objects, that is, there is a reverse-path buffer, a forward-path buffer, and a mail data buffer. Specific commands cause information to be appended to a specific buffer, or cause one or more buffers to be cleared. In accordance with the SMTP Pipelining model, all commands in a transmission by the client must be matched by responses from the server, or the HMTP transaction is invalid.

F.4.3.1.1 HELLO w/ EXTENSIONS (EHLO)

The EHLO command is used to identify the HMTP client to the HMTP server and that the use of SMTP extensions is required. The argument field contains the host name of the HMTP client.

The HMTP server identifies itself to the HMTP client in the connection greeting reply, and in the response to this command.

This command and an OK reply to it confirm that both the HMTP client and the HMTP server are in the initial state, that is, there is no transaction in progress and all state tables and buffers are cleared.

F.4.3.1.2 MAIL (MAIL)

The MAIL command is used to initiate a mail transaction in which the mail data is delivered to one or more mailboxes. The argument field contains a reverse-path.

The reverse-path consists of the sender mailbox. When a list of hosts is present it is a "reverse" source route and indicates that the mail was relayed through each host on the list (the first host in

the list was the most recent relay). This list is used as a source route to return non-delivery notices to the sender. As each relay host adds itself to the beginning of the list, it must use its name as known in the mail transport system (MTS) to which it is relaying the mail rather than the MTS from which the mail came (if they are different). In some types of error reporting messages (for example, undeliverable mail notifications) the reverse-path may be null.

This command clears the reverse-path buffer, the forward-path buffer, and the mail data buffer; and inserts the reverse-path information from this command into the reverse-path buffer.

F.4.3.1.3 RECIPIENT (RCPT)

The RCPT command is used to identify an individual recipient of the mail data; multiple recipients are specified by multiple use of this command.

The forward-path consists of a required destination mailbox. Source routing shall not be used.

When mail is relayed, the relay host must put itself at the beginning of the reverse-path. When mail reaches its ultimate destination, the HMTP server inserts it into the destination mailbox in accordance with its host mail conventions.

F.4.3.1.4 DATA (DATA)

The server treats the lines following the DATA command as mail data from the sender. This command causes the mail data from this command to be appended to the mail data buffer. The mail data may contain any of the 128 ASCII character codes.

The mail data is terminated by a line containing only a period, that is the character sequence "<CRLF>.<CRLF>" (see Section 4.4.2 on Transparency). This is the end of mail data indication.

The end-of-mail-data indication requires that the server must now process the stored mail transaction information. This processing consumes the information in the reverse-path buffer, the forward-path buffer, and the mail data buffer, and on the completion of this command these buffers are cleared. If the processing is successful the server must send an OK reply. If the processing fails completely the server must send a failure reply.

When the HMTP server accepts a message either for relaying or for final delivery it inserts at the beginning of the mail data a time stamp line. The time stamp line indicates the identity of the host that sent the message, and the identity of the host that received the message (and is inserting this time stamp), and the date and time the message was received. Relayed messages will have multiple time-stamp lines.

When the HMTP server makes the "final delivery" of a message it inserts at the beginning of the mail data a return path line. The return path line preserves the information in the <reverse-path> from the MAIL command. Here, final delivery means the message leaves the HMTP world. Normally, this would mean it has been delivered to the destination user, but in some cases it may be further processed and transmitted by another mail system.

It is possible for the mailbox in the return path be different from the actual sender's mailbox, for example, if error responses are to be delivered a special error handling mailbox rather than the message senders.

The preceding two paragraphs imply that the final mail data will begin with a return path line, followed by one or more time stamp lines. These lines will be followed by the mail data header and body [2]. See Example 8.

Special mention is needed of the response and further action required when the processing following the end of mail data indication is partially successful. This could arise if after accepting several recipients and the mail data, the HMTP server finds that the mail data can be successfully delivered to some of the recipients, but it cannot be to others (for example, due to mailbox space allocation problems). In such a situation, the response to the DATA command must be an OK reply. But, the HMTP server must compose and send an "undeliverable mail" notification message to the originator of the message. Either a single notification which lists all of the recipients that failed to get the message, or separate notification messages must be sent for each failed recipient (see Example 7). All undeliverable mail notification messages are sent using the MAIL command.

F.4.3.1.5 RESET (RSET)

The RSET command specifies that the current mail transaction is to be aborted. Any stored sender, recipients, and mail data must be discarded, and all buffers and state tables cleared. The server must send an OK reply.

F.4.3.1.6 QUIT (QUIT)

The QUIT command specifies that the server must send an OK reply, and then close the transmission channel.

The server should not close the transmission channel until it receives and replies to a QUIT command (even if there was an error). The client should not close the transmission channel until it send a QUIT command and receives the reply (even if there was an error response to a previous command). If the connection is closed prematurely the server should act as if a RSET command had been received (canceling any pending transaction, but not undoing any previously completed transaction), the client should act as if the command or transaction in progress had received a temporary error (4xx).

F.4.3.2 HMTP REPLIES

Replies to HMTP commands are devised to ensure the synchronization of requests and actions in the process of mail transfer, and to guarantee that the HMTP client always knows the state of the HMTP server. Every command must generate exactly one reply. In accordance with existing specifications for SMTP Command Pipelining, each command in a group must be matched to a reply in a group

An HMTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is meant for the human user. It is intended that the three digits contain enough encoded information that the HMTP client need not examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent and context dependent, so there are likely to be varying texts for each reply code. A discussion of the theory of reply codes can be found in Appendix E to RFC 822. Formally, a reply is defined to be the sequence: a three-digit code, <SP>, one line of text, and <CRLF>.

F.4.3.2.1 Error and Warning Codes

500 Syntax error, command unrecognized [This may include errors such as command line too long]
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented
552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed [E.g., mailbox syntax incorrect]
550 Requested action not taken: mailbox unavailable [E.g., mailbox not found, no access]
554 Transaction failed

F.4.3.2.2 Service-Management Codes

220 <domain> HMTP Service ready
221 <domain> Service closing transmission channel
250 Requested mail action okay, completed

354 Start mail input; end with <CRLF>.<CRLF>

421 <domain> Service not available, closing transmission channel [This may be a reply to any command if the service knows it must shut down]

450 Requested mail action not taken: mailbox unavailable [E.g., mailbox busy]
451 Requested action aborted: error in processing
452 Requested action not taken: insufficient system storage

F.4.4 TRANSPARENCY

Without some provision for data transparency the character sequence "<CRLF>.<CRLF>" ends the mail text and cannot be sent by the user. In general, users are not aware of such "forbidden" sequences. To allow all user composed text to be transmitted transparently the following procedures are used.

1. Before sending a line of mail text the sender-HMTP checks the first character of the line. If it is a period, one additional period is inserted at the beginning of the line.

2. When a line of mail text is received by the receiver-HMTP it checks the line. If the line is composed of a single period it is the end of mail. If the first character is a period and there are other characters on the line, the first character is deleted.

The mail data may contain any of the 128 ASCII characters. All characters are to be delivered to the recipient's mailbox including format effectors and other control characters. If the transmission channel provides an 8-bit byte (octets) data stream, the 7-bit ASCII codes are transmitted right justified in the octets with the high order bits cleared to zero.

In some systems it may be necessary to transform the data as it is received and stored. This may be necessary for hosts that use a different character set than ASCII as their local character set, or that store data in records rather than strings. If such transforms are necessary, they must be reversible -- especially if such transforms are applied to mail being relayed.

Transparency for arbitrary binary data embedded or attached to the HMTP mail message shall be accomplished in accordance with the applicable Internet Standards and RFCs for Multimedia Internet Mail Extensions (MIME).

F.4.5 Subnetwork Service Requirements for HMTP

Clients and Servers for the HF Mail Transfer Protocol bind to the HF Subnetwork at SAP ID 3. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

The commands, replies, and message data associated with the HF Mail Transfer Protocol are submitted to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

The address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the host on which the HMTP client or server is located.

The encoded data for the commands, replies, and user message data for the HF Mail Transfer Protocol are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) or each character aligned with the LSB of the octet. Message data encoded as seven-bit quantities (as in earlier messaging systems) shall be bit-aligned, LSB to LSB, with the octets of the S_Primitive, and the eighth (i.e, MSB) of the octet set to zero.

F.5 POP3 FOR USE OVER STANAG 5066 TRANSPORT (HFPOP)

This section of this Annex presents an adaptation of the POP3 protocol, as described in Internet Standard 53 / RFC 1939 [4], for use over the HF subnetwork described in the main body of and mandatory annexes to this STANAG.

The presumption and requirement for an HFPOP client and server is that they use the HF subnetwork protocol stack directly, without intervening transport or network protocol. POP3 mail clients/servers that incorporate TCP/IP protocols in their connection may use the HF subnetwork, but they interface to the subnetwork as an IP/PPP client rather than as an HMTP client as described here.

Parts of RFC 1939 are reproduced here, sufficient to identify the changes made in defining the HFPOP service. This presentation however assumes reader familiarity with the RFC, as many requirements are referenced rather than restated.

F.5.1 Introduction

It will in general be impractical to maintain a message transport system (MTS) over HF radio and to allow immediate delivery of messages to a node when that node relies on HF radio for connectivity. It is still necessary to be able to manage the transfer of mail to these smaller nodes. To solve this problem, a node that can support an MTS entity offers a maildrop service to these less endowed nodes. The Post Office Protocol - Version 3 (POP3), as modified in this Annex, may be used to permit a client workstation to dynamically access a maildrop on a server-host over HF radio. Usually, this means that the POP3 protocol is used to allow a node to retrieve mail that the server is holding for it. This annex presents a revision of the POP3 protocol that is intended to provide efficient operation over HF radio links. This revised protocol will be referred to as "HFPOP" in this annex.

For the remainder of this annex, the term "client host" refers to a host making use of the HFPOP service, while the term "server host" refers to a host that offers the HFPOP service.

This annex does not specify how a client host enters mail into the transport system, although a method consistent with the philosophy of this annex entails client use of the HMTP protocol to send its mail to the remote maildrop and message transfer agent (MTA) at the server.

F.5.2 Basic Operation

Initially, the server host starts the HFPOP service by listening for incoming connections over the HF subnet. When a client host wishes to make use of the service, it establishes a connection with the server host. When the connection is established, the HFPOP server sends a greeting. The client and POP3 server then exchange commands and responses (respectively) until the connection is closed or aborted.

[4] Myers, J, M. Rose., "Post Office Protocol – Version 3", Carnegie Mellon (Myers), Dover Beach Consulting Inc. (Rose), Network Working Group Request for Comments, RFC 1939, May 1996

F.5.2.1 State Machine Overview

The client/server dialog is governed by a state-machine, as defined in Table F-3; these states are the equivalent in the POP3 and HFPO3 models.

Table F-3: HFPOP/POP3 States

State	Comment
UNCONNECTED	HF use as defined in RFC 1939; client and server have no connection established between them; state transitions: - to the AUTHORIZATION state when a connection is established between the client and server;
AUTHORIZATION	HF use as defined in RFC 1939; the server validates client authorization to access the POP mailbox service; state transitions: - to the TRANSACTION state if authorization succeeds, - remains in AUTHORIZATION state if authorization fails; - to UPDATE state if QUIT command received, timeout occurs, or the connection fails between client and server;
TRANSACTION	HF use as defined in RFC 1939; clients may determine mailbox status, download messages, and delete messages; state transitions: - to the UPDATE STATE if QUIT command received or if connection is lost or if connection is inactive; - remains in TRANSACTION state otherwise
UPDATE	HF use as defined in RFC 1939; the server finalizes all irrevocable requests (e.g., message deletion) made by a client that used the QUIT command prior to closing the connection and returning to the unconnected state; no irrevocable actions taken if the client-server connection terminated abnormally; state transitions: - to the UNCONNECTED state on completion of all UPDATE state actions.

F.5.2.2 HFPOP Command Overview

As in the case of HMTP and SMTP, commands in the HFPOP consist of a case-insensitive keyword, possibly followed by one or more arguments. All commands are terminated by a CRLF pair. Keywords and arguments consist of printable ASCII characters. Keywords and arguments are each separated by a single SPACE character. Keywords are three or four characters long. Each argument may be up to 40 characters long. Commands are given in Table F-4, with comment on their use and form in RFC 1939 as compared to their required use in HFPOP.

Table F-4 - Summary of HFPOP/POP3 Commands

HFPOP Command	POP3 Command	HF POP use	Comment
Commands valid in the AUTHORIZATION state			
USER name	USER name	optional[(a)]	***Use strongly discouraged in HFPOP*** in favor of the APOP command; if used, should be pipelined with the PASS command.
PASS string	PASS string	optional[(a)]	***Use strongly discouraged in HFPOP*** in favor of the APOP command; if used, should be pipelined with the USER command.
APOP name digest	<*same*>	required[(b)]	Preferred/required form of client authentication
QUIT	<*same*>	required	HFPOP use as specified in RFC1939; sent by client for graceful exit from failed authorization attempt
Commands valid in the TRANSACTION state			
STAT	<*same*>	optional	HFPOP use as specified in RFC1939;
LIST [msg]	<*same*>	required	HFPOP use as specified in RFC1939;
RETR [msg] (message number is optional)	RETR msg (message number is required)	required	***HFPOP use modifies RFC1939***; all messages in the mail box can be retrieved in HFPOP by omitting the message number; RFC1939 requires a message number, and the command acts only on the specified message
DELE msg	<*same*>	required	HFPOP use as specified in RFC1939; the designated message is deleted from the HFPOP mailbox;
NOOP	<*same*>	optional	required by RFC1939;optional in HFPOP; no operation taken; may be used as fill in an HF subnetwork primitive
RSET	<*same*>	required	HFPOP use as specified in RFC1939;
TOP [msg n]	TOP msg n	required	***HFPOP use modifies RFC1939***; the tops of all messages in the mail box can be retrieved in HFPOP by omitting the message number; RFC1939 requires a message number, and the command acts only on the specified message
UIDL [msg]	<*same*>	optional	HFPOP use as specified in RFC1939; retrieves the Unique IDentifier List (UIDL) for the specified message, or for all messages if the message number is omitted
QUIT	<*same*>	required	HFPOP use as specified in RFC1939; terminates the TRANSACTION state and forces transition to the UPDATE state

Notes: (a) use of the USER and PASS commands is required for minimal client implementations in RFC1939; it is not required for the HFPOP service; (b) use of the APOP command (which is optional in RFC1939) is recommended as the more efficient method of user authentication in the HF POP protocol.

Responses in the HFPOP service consist of a status indicator and a keyword possibly followed by additional server-dependent information. A CRLF pair terminates all responses. Responses may be up to 512 characters long, including the terminating CRLF. There are two status indicators: positive ("+OK") and negative ("-ERR"). Servers shall send the "+OK" and "-ERR" in upper case. With the exception of the STAT, LIST, RETR, TOP, and UIDL commands the server response is significant only to the “+OK” and “-ERR”. Any text occurring after these replies may be ignored by the client.

Responses to certain commands are multi-line. In these cases, which are clearly indicated below, after sending the first line of the response and a CRLF, any additional lines are sent, each terminated by a CRLF pair. When all lines of the response have been sent, a final line is sent, consisting of a termination octet (decimal code 046, ".") and a CRLF pair. If any line of the multi-line response begins with the termination octet, the line is "byte-stuffed" by pre-pending the termination octet to that line of the response. Hence a multi-line response is terminated with the five octets "CRLF.CRLF". When examining a multi-line response, the client checks to see if the line begins with the termination octet. If so and if octets other than CRLF follow, the first octet of the line (the termination octet) is stripped away. If so and if CRLF immediately follows the termination character, then the response from the POP server is ended and the line containing ".CRLF" is not considered part of the multi-line response. Note that the HFPOP procedures for data transparency are similar to those defined for the HMTP. Separate definitions are provided in this annex because the parent documents which define these procedures are different (i.e., RFC 821 defines the data transparency requirements for HMTP/SMTP and RFC 1939 defines the data transparency requirements for HFPOP/POP3).

A HFPOP session progresses through a number of states during its lifetime. Once the SnAP connection has been opened and the HFPOP server has sent the greeting, the session enters the AUTHORIZATION state. In this state, the client must identify itself to the HFPOP server. Once the client has successfully done this, the server acquires resources associated with the client's maildrop, and the session enters the TRANSACTION state. In this state, the client requests actions on the part of the HFPOP server. When the client has issued the QUIT command, the session enters the UPDATE state. In this state, the HFPOP server releases any resources acquired during the TRANSACTION state and says goodbye. The connection is then closed.

A server MUST respond to an unrecognized, unimplemented, or syntactically invalid command by responding with a negative status indicator. A server MUST respond to a command issued when the session is in an incorrect state by responding with a negative status indicator. There is no general method for a client to distinguish between a server that does not implement an optional command and a server that is unwilling or unable to process the command.

A HFPOP server MAY have an inactivity autologout timer. Such a timer MUST be of at least 10 minutes' duration. The receipt of any command from the client during that interval should suffice to reset the autologout timer. When the timer expires, the session does NOT enter the UPDATE state--the server should close the connection without removing any messages or sending any response to the client.

F.5.3 The AUTHORIZATION State

Once the connection has been opened by a HFPOP client, the HFPOP server shall issue a one line greeting as follows:

S: +OK HFPOP (STANAG 5066) server ready <process-ID.clock@hostname>

An HF_POP3 client shall interpret responses that do not conform to this format as negative responses. The HFPOP session is now in the AUTHORIZATION state. The HFPOP client must now identify and authenticate itself to the HFPOP server. The APOP command shall be used for this purpose in preference over the USER/PASS command pair.

Once the POP3 server has determined through the use of the APOP command that the client should be given access to the appropriate maildrop, the HFPOP server then acquires an exclusive-access lock on the maildrop, as necessary to prevent messages from being modified or removed before the session enters the UPDATE state. If the lock is successfully acquired, the POP3 server responds with a positive status indicator plus the response to the LIST command with no arguments (described in the following section on the TRANSACTION state). The HFPOP session now enters the TRANSACTION state, with no messages marked as deleted. If the maildrop cannot be opened for some reason (for example, a lock can not be acquired, the client is denied access to the appropriate maildrop, or the maildrop cannot be parsed), the HFPOP server responds with a negative status indicator. (If a lock was acquired but the HFPOP server intends to respond with a negative status indicator, the HFPOP server must release the lock prior to rejecting the command.) After returning a negative status indicator, the server may close the connection. If the server does not close the connection, the client may either issue a new authentication command and start again, or the client may issue the QUIT command.

After the HFPOP server has opened the maildrop, it assigns a message- number to each message, and notes the size of each message in octets. The first message in the maildrop is assigned a message-number of "1", the second is assigned "2", and so on, so that the nth message in a maildrop is assigned a message-number of "n". In HFPOP commands and responses, all message-numbers and message sizes are expressed in base-10 (i.e., decimal).

If the HFPOP client does not receive a positive response from the server, either to the APOP command, USER/PASS command pair, or in the initial server greeting, the client may exit the AUTHORIZATION state using the QUIT command. Alternatively, it may try again, or close the subnetwork connection. Graceful exit using the QUIT command is the preferred method for terminating a failed authorization attempt.

The basic HFPOP commands are described below, quoting extensively from RFC 1939.

F.5.3.1 APOP Command

Command Format: APOP *name digest*

Arguments: *name* - a string identifying a mailbox and,
digest - a MD5 digest string (both required)

Restrictions: may only be given in the AUTHORIZATION state after the HFPOP greeting.

In the context of this standard, the APOP command provides both origin authentication and replay protection. An HFPOP server shall include a timestamp in its banner greeting. The syntax of the timestamp corresponds to the `msg-id' in [RFC822], and MUST be different each time the POP3 server issues a banner greeting. For example, on a UNIX implementation in which a separate UNIX process is used for each instance of a POP3 server, the syntax of the timestamp might be:

<process-ID.clock@hostname>

where `process-ID' is the decimal value of the process's PID, clock is the decimal value of the system clock, and hostname is the fully-qualified domain-name corresponding to the host where the POP3 server is running.

The POP3 client makes note of this timestamp, and then issues the APOP command. The `name' parameter has identical semantics to the `name' parameter of the USER command. The `digest' parameter is calculated by applying the MD5 algorithm [RFC1321] to a string consisting of the timestamp (including angle-brackets) followed by a shared secret. This shared secret is a string known only to the POP3 client and server. Great care should be taken to prevent unauthorized disclosure of the secret, as knowledge of the secret will allow any entity to successfully masquerade as the named user. The `digest' parameter itself is a 16-octet value that is sent in hexadecimal format, using lower-case ASCII characters.

When the POP3 server receives the APOP command, it verifies the digest provided using the MD5 algorithm. If the digest is correct, the POP3 session enters the TRANSACTION state and the POP3 server issues a positive response including the LIST response for the maildrop, as shown below. Otherwise, a negative response is issued and the HFPOP session remains in the AUTHORIZATION state.

Note that as the length of the shared secret increases, so does the difficulty of deriving it. As such, shared secrets should be long strings (considerably longer than the 8-character example shown below).

Possible Responses:
S: +OK maildrop locked and ready; maildrop has 1 message (320 octets)
S: 1 120
S: 2 200
S: .

S: -ERR permission denied

Examples:
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)
S: 1 369

In this example, the shared secret is the string `tanstaaf'. Hence, the MD5 algorithm is applied to the string

<1896.697170952@dbc.mtview.ca.us>tanstaaf

which produces a digest value of

c4c9334bac560ecc979e58001b3e22fb

F.5.3.2 USER and PASS Commands

Command Format: USER *username*

Arguments: *username* - a string identifying a username (and associated mailbox)

Command Format: PASS *password_string*

Arguments: *password_string* - a string identifying the password for the mailbox/user name given in the preceding USER command

Restrictions: may only be given in the AUTHORIZATION state after the POP3 greeting.

The USER and PASS commands are defined in RFC1939 as the means by which a minimal implementation of the POP3 service determines if a connecting client is authorized to access the mailbox designated for the named user. Their use in HFPOP is optional, but strongly discouraged, because of potential inefficiencies when used in an HF subnetwork. These inefficiencies arise if the USER command and its companion PASS command are used interactively and sent in different transmission intervals, rather than pipelined and sent to the server in the same HF transmission.

If the USER and PASS commands are implemented and used in the HF POP service, the USER and PASS commands should be pipelined, i.e., sent by the client together in sequence without the client waiting for an initial server response to the USER command.

F.5.3.3 QUIT Command

Command Format: QUIT

Arguments: none

Restrictions: none

In the AUTHORIZATION state, a client may use the QUIT command to exit after a failed authorization attempt, as a means of gracefully closing the connection. Use of the QUIT command by the HFPOP service shall be as defined in RFC1939.

Possible Responses:
+OK

Examples:
C: QUIT
S: +OK dewey POP3 server signing off

F.5.4 The TRANSACTION State

Once the client has successfully identified itself to the POP3 server and the POP3 server has locked and opened the appropriate maildrop, the POP3 session is now in the TRANSACTION state. Note that this transition occurs after the APOP line from the client is processed and before the server generates the response for any subsequent commands pipelined in the same HF transmission interval as the APOP command. The client may now issue POP3 commands repeatedly. After each command, the POP3 server issues a response. Note that while client commands may be pipelined in the same S_UNIDATA_REQUEST primitive or transmission interval, it may be possible for the HFPOP server responses to be submitted for transmission to the client before all commands have been received. Eventually, the client sends the QUIT command and the POP3 session enters the UPDATE state.

Typical POP3 commands valid in the TRANSACTION state (though not all of them) are listed in the paragraphs below. It should be noted that a few of these have been modified to facilitate command pipeling for the HFPOP service and represent non-standard use of the RFC1939 command set. These modified commands have been noted.

F.5.4.1 LIST Command

Command Format: LIST *[msg]*

Arguments: *msg* - a message-number (optional), that, if present, may NOT refer to a message marked as deleted

Restrictions: may only be given in the TRANSACTION state

If an argument was given and the POP3 server issues a positive response with a line containing information for that message. This line is called a "scan listing" for that message.

If no argument was given and the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, for each message in the maildrop, the POP3 server responds with a line containing information for that message. This line is also called a "scan listing" for that message. If there are no messages in the maildrop, then the POP3 server responds with no scan listings--it issues a positive response followed by a line containing a termination octet and a CRLF pair.

In order to simplify parsing, all POP3 servers are required to use a certain format for scan listings. A scan listing consists of the message-number of the message, followed by a single space and the exact size of the message in octets, followed by a CRLF pair. Methods for calculating the exact size of the message are described in the "Message Format" section of RFC 1939.

Note that messages marked as deleted are not listed.

Possible Responses:
+OK scan listing follows
-ERR no such message

Examples:
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
...
C: LIST 2
S: +OK 2 200
...
C: LIST 3
S: -ERR no such message, only 2 messages in maildrop

F.5.4.2 RETR (Retrieve) Command

Command Format: RETR [*nn*]

Arguments: *nn* a message-number (optional) that may NOT refer to a message marked as deleted {Note that this is a modification of the RFC 1939, for which the message number is required).

Restrictions: may only be given in the TRANSACTION state

This command has been modified from the definition of RFC 1939 so that the message number is optional. When omitted, the RETR command retrieves all messages in the mail drop, otherwise the designated message is retrieved. This message was modified to allow a client to compose a short message of pipelined HFPOP commands and retrieve all messages in the maildrop. Even with command pipelining, use of the standards of RFC 1939 would have required a client to access the maildrop and request a listing of the number of messages and then send a RETR message for each. The modified RETR command with optional message number field is considered a more efficient approach that does not preclude the use of the standard.

If the POP3 server issues a positive response, then the response given is multi-line. If an argument was given, after the initial +OK, the POP3 server sends the message corresponding to the given message-number, being careful to byte-stuff the termination character (as with all multi-line responses). If no argument was given, the server sends all messages in the drop that are not marked as deleted. Message boundaries in multi-message responses are the same as for single messages. Each message is bounded initially by a +OK response, and finally by the termination octet (decimal code 046, "."), with the message byte-stuffed for data transparency as required.

Possible Responses:
+OK message nn follows
+OK m messages follow
-ERR no such message

Examples:
single message retrieval
C: RETR 1

S: +OK 120 octets
S: <the HFPOP server sends the entire message here, with data transparency as required>
S: .

mutli-message retrieval, two messages of 120 and 289 bytes respectively.
C: RETR
S: +OK 120 octets
S: <*the HFPOP server sends the first message here*>
S: .
S: +OK 289 octets
S: <*the HFPOP server sends the second message here* >
S: .

F.5.4.3 DELE (Delete) Command

Command Format: DELE *msg*

Arguments: *msg* - a message-number (required) that may NOT refer to a message marked as deleted.

Restrictions: may only be given in the TRANSACTION state

This command is used in HFPOP as specified in RFC 1939. The POP3 server marks the message as deleted. Any future reference to the message-number associated with the message in a POP3 command generates an error. The POP3 server does not actually delete the message until the POP3 session enters the UPDATE state.

Possible Responses:
+OK message deleted
-ERR no such message

Examples:
C: DELE 1
S: +OK message 1 deleted
...
C: DELE 2
S: -ERR message 2 already deleted

F.5.4.4 RSET (Reset) Command

Command Format: RSET

Arguments: none

Restrictions: may only be given in the TRANSACTION state

This command is used in HFPOP as specified in RFC 1939. If any messages have been marked as deleted by the POP3 server, they are unmarked. The POP3 server then replies with a positive response.

Possible Responses:
+OK

Examples:
C: RSET
S: +OK maildrop has 2 messages (320 octets)

F.5.4.5 STAT (Status) Command

Command Format: STAT

Arguments: none

Restrictions: may only be given in the TRANSACTION state

This command is used in HFPOP as specified in RFC 1939. The POP3 server issues a positive response with a line containing information for the maildrop. This line is called a "drop listing" for that maildrop.

In order to simplify parsing, all POP3 servers are required to use a certain format for drop listings. The positive response consists of "+OK" followed by a single space, the number of messages in the maildrop, a single space, the size of the maildrop in octets, and the line shall be terminated with a CRLF pair.

Note that messages marked as deleted are not counted in either total.

Possible Responses:
+OK nn mm

Examples:
C: STAT
S: +OK 2 320

F.5.4.6 TOP Command (Optional)

Command Format: TOP *[msg n]* *or* TOP *[n]*

Arguments: *msg* - a message-number that may NOT refer to to a message marked as deleted, and
n - a non-negative number of lines; these are optional arguments
[Note: this is a modification of the provisions of RFC 1939, where these parameters are required];

Restrictions: may only be given in the TRANSACTION state

The message was modified for HFPOP to allow omission of the message number and numer of lines to be retrieved. The reason for modification was the same reason for modifying the RETR command, to allow more efficient pipeling of commands when a node intially access its mail box and cannot know how many messages are present. If both parameters are omitted in this command, the HFPOP server shall respond with the headers for all messages in the mail box. If only one parameter is present, it shall be interpreted as the number of lines parameter, and the HFPOP server shall respond by sending the headers and the number of lines requested for all messages in the mail drop. If the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, the POP3 server sends the headers of the message, the blank line separating the headers from the body, and then the number of lines of the indicated message's body, being careful to byte-stuff the termination character (as with all multi-line responses).

Note that if the number of lines requested by the POP3 client is greater than the number of lines in the body, then the POP3 server sends the entire message. The TOP command and its modified form presented here allow a client to download information for all messages in the maildrop, without first knowing the number or sizes of the messages or clogging the link by unwitting transmission of very large files.

Possible Responses:
+OK top of message follows
+OK tops of m messages follow
-ERR no such message

Examples:

top of a single message retrieved

C: TOP 1 10
S: +OK
S: <the POP3 server sends the headers of the message, a blank line, and the first 10 lines of the body of the message>
S: .
...

invalid message number used

C: TOP 100 3
S: -ERR no such message

tops of a three messages retrieved

C: TOP 5
S: +OK
S: <the POP3 server sends first message header, a blank line, and 5 lines of the body >
S: .
S: +OK
S: <the POP3 server sends second message header, a blank line, and 5 lines of the body >
S: .
S: +OK
S: <the POP3 server sends third message header, a blank line, and 5 lines of the body >
S: .
...

F.5.4.7 UIDL (Unique Identifier List) Command (Optional)

Command Format: UIDL *[msg]*

Arguments: *msg* - a message-number (optional), that, if present, may NOT refer to a message marked as deleted; optional use of the message number is defined in RFC 1939.

Restrictions: may only be given in the TRANSACTION state.

If an argument was given and the POP3 server issues a positive response with a line containing information for that message. This line is called a "unique-id listing" for that message.

If no argument was given, then the response given is multi-line. After the initial +OK, for each message in the maildrop, the HFPOP server responds with a line containing information for that message. This line is called a "unique-id listing" for that message.

In order to simplify parsing, all POP3 servers are required to use a certain format for unique-id listings. A unique-id listing consists of the message-number of the message, followed by a single space and the unique-id of the message. No information follows the unique-id in the unique-id listing.

The unique-id of a message is an arbitrary server-determined string, consisting of one to 70 characters in the range 0x21 to 0x7E, that uniquely identifies a message within a maildrop and that persists across sessions. This persistence is required even if a session ends without entering the UPDATE state. The server should never reuse an unique-id in a given maildrop, for as long as the entity using the unique-id exists.

Note that messages marked as deleted are not listed.

While it is generally preferable for server implementations to store arbitrarily assigned unique-ids in the maildrop, this specification is intended to permit unique-ids to be calculated as a hash of the message. Clients should be able to handle a situation where two identical copies of a message in a maildrop have the same unique-id.

Possible Responses:
+OK unique-id listing follows
-ERR no such message

Examples:
C: UIDL
S: +OK
S: 1 whqtswO00WBw418f9t5JxYwZ
S: 2 QhdPYR:00WBw1Ph7x7
S: .
...
C: UIDL 2
S: +OK 2 QhdPYR:00WBw1Ph7x7
...
C: UIDL 3
S: -ERR no such message, only 2 messages in maildrop

F.5.5 The UPDATE State

As for POP3 as defined in RFC 1939, when the client issues the QUIT command from the TRANSACTION state, the HFPOP session enters the UPDATE state. Note that if the client issues the QUIT command from the AUTHORIZATION state, the HFPOP session terminates but does NOT enter the UPDATE state. If a session terminates for any reason other than a client-issued QUIT command during the TRANSACTION state, the HFPOP session does NOT enter the UPDATE state and MUST not remove any messages from the maildrop.

When the UPDATE state is entered from the TRANSACTION state after a client-issued QUIT command, the HFPOP server removes all messages marked as deleted from the maildrop and replies as to the status of this operation. If there is an error, such as a resource shortage, encountered while removing messages, the maildrop may reach an end state where some or none of the messages marked as deleted are removed. In no case may the server remove any messages not marked as deleted.

Whether the removal was successful or not, the server then releases any exclusive-access lock on the maildrop and closes the connection.

Possible Responses:
+OK
-ERR some deleted messages not removed

Examples:
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
...
C: QUIT
S: +OK dewey POP3 server signing off (2 messages left)
...

F.5.6. Scaling, Operational, and Performance Considerations

As a server-specific configuration option, the HFPOP protocol may implement a special case of a site policy in order to prevent users from accumulating large mail queues on the server system. These large queues can result in a full disk and system crashes; this problem is frequently observed in POP3 servers. Messages may only be downloaded once from the server, and shall be deleted after this has been accomplished. This could be implemented in HFPOP server software by the following mechanism: "following a POP3 login by a client that was ended by a QUIT, delete all messages downloaded during the session with the RETR command". It is important not to delete messages in the event of abnormal connection termination (ie, if no QUIT was received from the client) because the client may not have successfully received or stored the messages. Servers implementing a download-and-delete policy may also wish to disable or limit the optional TOP command, since it could be used as an alternate mechanism to download entire messages.

With respect to command pipelining and performance, the reason for limited modifications to the POP3 commands has been presented in earlier discussion of the modifications to the RETR and TOP commands. An example of pipelined-use of these commands is given below:

connection, access authorization, retrieval of all three messages in a maildrop, and session termination:

S: +OK HFPOP server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
C: RETR
C: QUIT

S: +OK maildrop has 3 messages (4679 octets)
S: +OK
S: <the HFPOP server sends first message>
S: .
S: +OK
S: <the HFPOP server sends second message>
S: .
S: +OK
S: <the HFPOP server sends third message header>
S: .
S: +OK dewey HFPOP server signing off (no messages left)

F.5.7 Subnetwork Service Requirements for HFPOP

Clients and servers for the HFPOP service bind to the HF Subnetwork at SAP ID 4. Client rank and priority are implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

The commands, replies, and message data associated with the HFPOP Service are submitted to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

The address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the host on which the HFPOP client or server is located.

The encoded data for the commands, replies, and user message data for the HFPOP Service are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) or each character aligned with the LSB of the octet. Message data encoded as seven-bit quantities (as in earlier messaging systems) shall be bit-aligned, LSB to LSB, with the octets of the S_Primitive, and the eighth (i.e, MSB) of the octet set to zero.

F.6 Operator orderwire application

This section specifies the use of existing primitives for transfer of operator orderwire messages over a radio link using the HF subnet described in the mandatory part of this STANAG. These messages would generally be displayed on the system hosting HF-subnetwork protocol stack, and the subnet client would likewise reside on the same host as the node. This is an implementation decision and is described only as an example.

For acknowledged orderwire, the procedures described in section F.1 shall be used. For non-acknowledged orderwire, the procedures described in section F.2 shall be used. Orderwire clients shall attach to SAP ID 5.

F.7 Reliable Connection-Oriented Protocol

This subsection defines a simple Reliable Connection-Oriented Protocol (RCOP) for reliable data connections using the ARQ services of the HF subnetwork, with a minimal header to support multiplexed connections between a pair of nodes through a single hard or soft link.

F.7.1 RCOP Data Unit

All RCOP clients shall segment their U_PDUs submitted in S_UNIDATA_REQUEST primitives to comply with the size of the Maximum Transmission Unit (MTU) advertised by the subnetwork interface sublayer in the S_BIND_ACCEPTED primitive. The format of all RCOP U_PDUs shall be as shown in Figure F-1. Order of transmission of bits and bytes, and mapping of bits into fields, shall be as defined in the mandatory parts of this document.

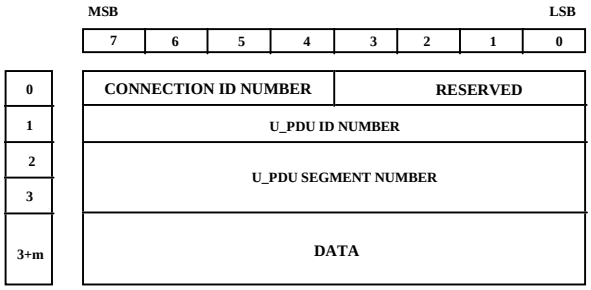

Figure F-1. Format for Reliable Connection-Oriented Protocol Data Units

The following are required for RCOP PDUs:

1. The connection ID number shall be a value from 0-15. Assignment and co-ordination of connection ID numbers is not specified here. Connection ID number 0 shall be reserved for non-multiplexed connections.
2. The reserved bits shall be set to 0.
3. The U_PDU ID numbers shall be assigned consecutively to U_PDUs.

4. The U_PDU segment number shall be assigned consecutively to segments within a single U_PDU. The first segment transmitted shall be assigned segment number 0. If a U_PDU is not segmented, the single segment that is transmitted shall be assigned number 0.

Note that, in accordance with the provisions of Annex A of this STANAG, if the subnetwork interface sublayer receives a S_UNIDATA_REQUEST primitive with a U_PDU larger than the maximum MTU size, the S_UNIDATA_REQUEST shall be rejected.

F.7.2 RCOP Subnetwork Service Requirements

RCOP clients bind to the HF Subnetwork at SAP ID 6. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

An RCOP client submits its PDUs to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY or AS_THEY_ARRIVE

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host at which the destination RCOP client is located. If AS_THEY_ARRIVE delivery-order is specified, the RCOP client at the destination may be responsible for reassembling the U_PDU segments in proper sequence order.

The encoded data for the RCOP client are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

F.7.3 Example Application and Extended Client Definition: A Basic File Transfer Protocol built on top of RCOP

As an example application of the reliable connection-oriented protocol, this section defines an interface specification for a Basic File Transfer Protocol (BFTP) which uses RCOP as support. This simple client definition is suitable for the transmission of large text or binary files whose size exceeds the advertised MTU size. This simple client and the RCOP protocol have been used to test and demonstrate client-level interoperability between different implementations of STANAG 5066 during its development.

F.7.3.1 BASIC FILE TRANSFER PROTOCOL DESCRIPTION

The basic FTP may be used to transfer files via the STANAG 5066 protocol. It is a very simple protocol and is not designed to be especially robust.

F.7.3.2 MESSAGE SPECIFICATION

The format for the basic file transfer protocol data unit (BFTP_PDU) is as follows :

PDU Header	Data (File for Transfer)

Figure F-2 - Basic FTP Protocol Data Unit (BFTP_PDU)

The BFTP header is described in further detail in Figure F-3 below:

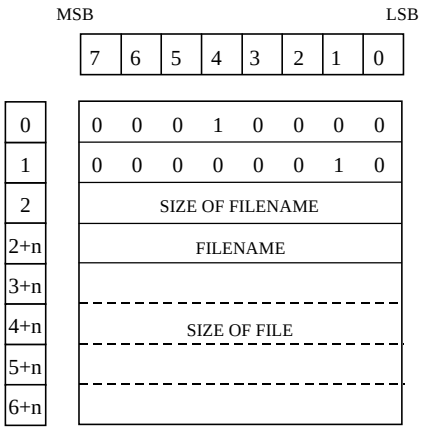

Figure F-3 - Header Frame for Protocol Data Unit

The first two bytes of the header are synchronization bytes. They represent the control bytes DLE (Data Link Escape) and STX (Start of Text).

The field SIZE OF FILENAME is a 1-byte fixed-length field that represents the number of bytes used for the FILENAME field.

The field FILENAME is a variable length field, the size of which is specified in the field SIZE OF FILENAME. This field represents the name of the file which is to be sent using the Basic File Transfer Protocol. The first byte of the filename shall be placed in the first byte of this field.

The field SIZE OF FILE is a 4 byte fixed length field which gives the size of the file to be sent. The first byte of this field shall be the highest order byte and the last byte the lowest order byte.

F.7.3.4 MESSAGE ACKNOWLEDGEMENT

On receiving the last byte of the message, the receiving client sends the two-octet reply message:

0x10 0x0B

to confirm that the entire message has been received. This is equivalent to the "ZEOF" message of the Z-modem protocol.

F.7.3.5 BFTP ENCAPSULATION within the RCOP PDU

The BFTP client encapsulates the BFTP_PDU in the RCOP_PDU data field. If the BFTP_PDU exceeds the maximum size of the data field in the RCOP_PDU (i.e, if the BTFP_PDU is larger than MTU_size – 4 octets), then the BFTP client segments the BTFP_PDU, placing successive segments in RCOP_PDUs with consecutive U_PDU sequence numbers. When received, the BFTP client reassembles the BFTP_PDU if it determines that the BFTP_PDU has been segmented. Subject to local-host file naming conventions, the BFTP client stores the received file with the name transmitted in the header with the file.

F.8 Unreliable Datagram-Oriented Protocol (UDOP)

This section defines a simple Unreliable Datagram-Oriented Protocol (UDOP) using the non-ARQ services of the HF subnetwork, with a minimal header to support multiplexed datagram delivery. Since non-ARQ services are used, the UDOP may support a multicast service through the use of group addresses within the HF Subnetwork.

F.8.1 UDOP Data Unit

UDOP Protocol Data Units shall be defined and used identically to those defined for the Reliable Connection-Oriented Protocol in section F.7.1.

F.8.2 UDOP Subnetwork Service Requirements

UDOP clients bind to the HF Subnetwork at SAP ID 7. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

A UDOP client submits its U_PDUs to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = non-ARQ
- Delivery Confirmation = none
- Deliver in Order = IN-ORDER DELIVERY or AS_THEY_ARRIVE

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host on which the destination message stores are located. If AS_THEY_ARRIVE delivery-order is specified, the UDOP client at the destination may be responsible for reassembling the U_PDU segments in proper sequence order.

The encoded data for the UDOP client are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

F.9 Point-to-Point Protocol (PPP) Client

The requirements for HF-subnetwork support of the Point-to-Point Protocol (PPP), Internet Request for Comments (RFC) 1548, are defined here. PPP is a well-defined and complex peer-

to-peer protocol whose requirements are not repeated here. The discussion of requirements below assumes familiarity with the contents of RFC 1548.

F.9.1 PPP Encapsulation

A PPP client shall construct PPP frames in accordance with Section 2 of RFC 1548, using the 16-bit Protocol-field to identify the information encapsulated within the PPP and its upper-layer source. No framing characters (e.g., the HDLC framing symbols) are included or allowed however, since the PPP packet is submitted to the HF subnetwork encapsulated in an S_UNIDATA_REQUEST Primitive and received from the subnetwork encapsulated in an S_UNIDATA_INDICATION Primitive. These S_Primitives provide all necessary framing information to mark the beginning and end of PPP frames. The two bytes of the PPP frame (i.e, the Protocol field) are placed in the first two bytes of the U_PDU field within the primitive, and so on to the last byte of the PPP frame and S_Primitive U_PDU field.

F.9.2 PPP Subnetwork Service Requirements

PPP clients bind to the HF Subnetwork at SAP ID 8. Client rank and priority are implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

All PPP frames are submitted to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host to which the PPP frame is being sent.

The encoded data of the PPP frame are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

Implementors of PPP clients are encouraged to make maximum use of the data compression capabilities of the Point-to-Point Protocol as a means of promoting HF subnetwork efficiency.

F.10 INTERNET Protocol Client

The requirements for HF-subnetwork support of the Internet Protocol, Internet Standard 5, Request for Comments (RFC) 791, are defined here. The Internet Protocol is a well known protocol whose requirements are not repeated here.

F.10.1 IP Encapsulation for Transmission within the HF Subnetwork

IP datagrams shall be contained within the U_DPU field of S_UNIDATA_REQUEST Primitives submitted to the HF subnetwork for transmission, and delivered to clients at the destination

node(s) within the U_DPU field of S_UNIDATA_INDICATION Primitives. There are no framing characters required or allowed. The first byte of the header of the IP datagram is aligned with the first byte of the U_PDU field within the primitive, and so on to the last byte of the datagram and U_PDU field.

F.10.2 IP Subnetwork Service Requirements

Clients using IP bind to the HF Subnetwork at SAP ID 9. Client rank and priority are implementation-dependent parameters for this application. Use of Rank = 15 is discouraged. IP support by the HF subnetwork may be configured to use either ARQ or non-ARQ delivery services on a case-by-case basis.

HF subnetwork support using reliable point-to-point delivery between a pair of nodes is preferred for efficiency in the IP and higher-layer protocols, but cannot support IP-multicast protocols. The service definition for reliable-IP datagram delivery shall be as follows:

- Transmission Mode = ARQ,
- Delivery Confirmation = NODE DELIVERY,
- Deliver in Order = IN-ORDER DELIVERY or AS-THEY-ARRIVE

The destination address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host to which the IP datagram is being sent. NOTE that this may NOT be the IP address contained within the datagram itself, because of the relay and routing properties of the IP protocol. Determination of the correspondence between HF subnetwork and IP addresses is beyond the scope of this STANAG, but may be accomplished through the use of the Address Resolution Protocol (ARP), Internet Standard 37 (RFC 826) or other mechanism.

If IP-multicast address groups must be supported within the HF subnetwork environment (i.e., an application wishes to take advantage of the broadcast nature of the HF channel to support multicast), then the HF subnetwork must be configured in non-ARQ mode to support this requirement. In such a case, the HF subnetwork service shall be configured as follows:

- Transmission Mode = non-ARQ,
- Delivery Confirmation = none,
- Deliver in Order = IN-ORDER DELIVERY or AS-THEY-ARRIVE
-

The number of repeats for the D_DPUs in the service may be set to a value greater than one to provide some increased probability of receipt and reliability when using the subnetwork for IP multicast support.

The encoded bytes of the IP datagram are bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.