

AD/A052 594  
**ARPANET PROTOCOL HANDBOOK**  
**NIC 7104, Rev. Jan. 1978**

**ARPANET**  
**PROTOCOL HANDBOOK**  
JANUARY 1978

Edited by:

ELIZABETH FEINLER  
Network Information Center  
SRI International  
Menlo Park, California 94025

And

JONATHAN POSTEL  
Information Sciences Institute  
University of Southern California  
4676 Admiralty Way  
Marina Del Rey, California 90291

Prepared for:

DEFENSE COMMUNICATIONS AGENCY  
Code 535, Washington, D. C. 20305

Published by:

Data Composition  
An Arcata National Company  
San Francisco, California

Approved for public release; distribution unlimited.

REPRODUCED BY  
**NATIONAL TECHNICAL**  
**INFORMATION SERVICE**  
U.S. DEPARTMENT OF COMMERCE  
SPRINGFIELD, VA. 22161



## UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NIC-7104-Rev-2	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ARPANET Protocol Handbook		5. TYPE OF REPORT & PERIOD COVERED
		6. PERFORMING ORG. REPORT NUMBER NIC-7104-Rev-2
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s) DCA200-C-641
9. PERFORMING ORGANIZATION NAME AND ADDRESS Network Information Center✓ SRI International Menlo Park, CA 94025		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Communications Agency Code 535 Washington, D. C. 20305		12. REPORT DATE Jan 78
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15.  Unclassified
16. DISTRIBUTION STATEMENT (of this Report)  Approved for Public Release, Distribution Unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES  Supercedes Report #NIC-7104-Rev-1, dated Apr 76, AD A027 964		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)  Communications networks, Computer networks, Message processing, Data transmission systems, On-line systems, modems, syntax, ARPANET, Protocols, Telnet system		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number)  The ARPANET Protocol Handbook is a collection of documents that describe the protocols currently in use on the ARPANET Computer Network as of January 1978. Protocols are the rules of communication between processes on the network. The protocols in use on the ARPANET form a tree structure. The basic protocol is the IMP-to-Host Protocol. Directly under that is the Host-to-Host Protocol. Spreading out beneath, but still closely related, are the process level protocols: TELNET, File Transfer, Remote Job Entry, and Graphics. (Cont'd.)		

UNCLASSIFIED

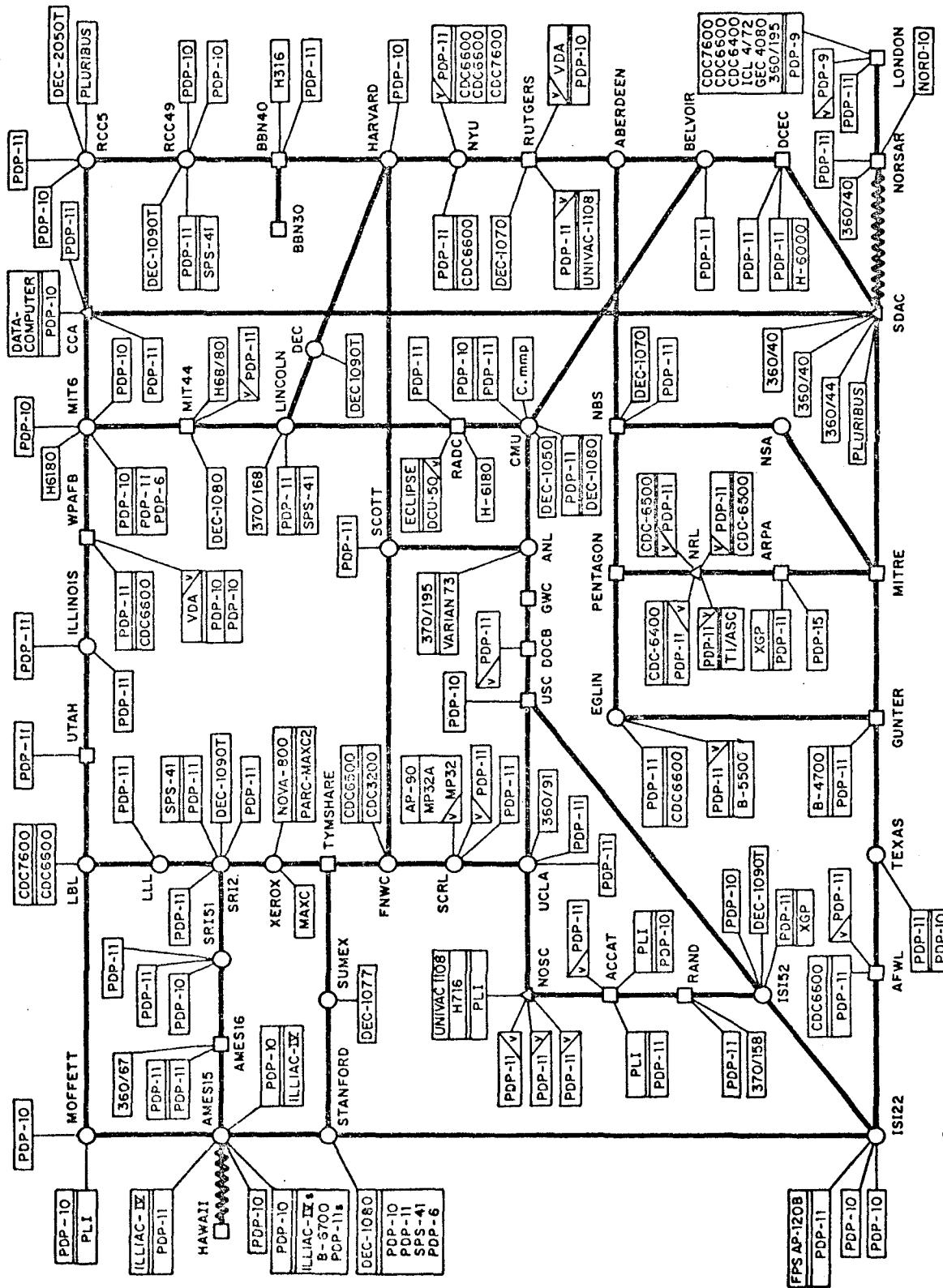
SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

Item #20 Cont'd:

Interspersed along the way are a few small protocols such as the Initial Connection Protocol, and the definition of the standard character set.

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

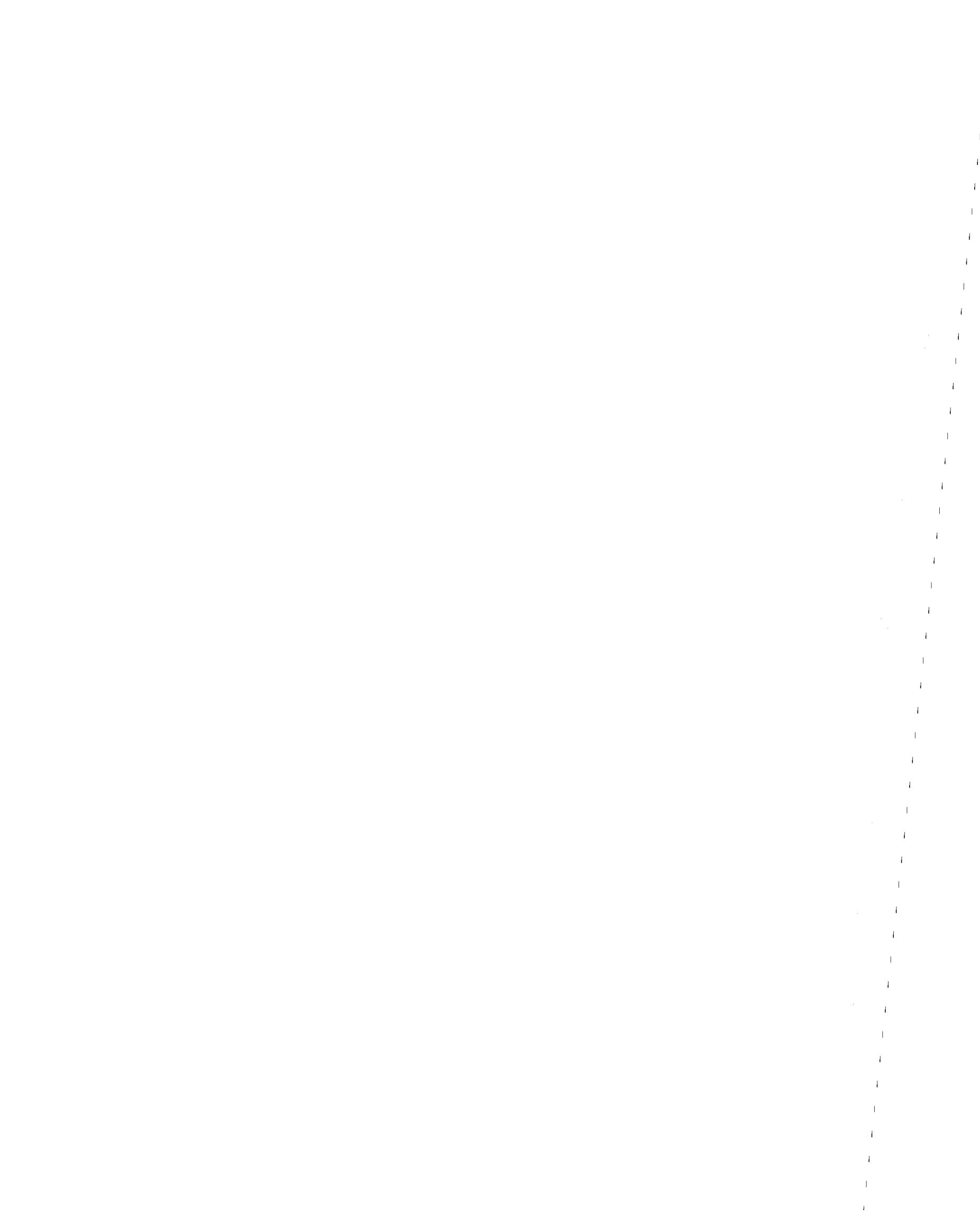
ARPANET LOGICAL MAP, SEPTEMBER 1977



**PLEASE NOTE THAT WHILE THIS MAP SHOWS THE HOST POPULATION OF THE NETWORK ACCORDING TO THE BEST INFORMATION OBTAINABLE, NO CLAIM CAN BE MADE FOR ITS ACCURACY. HOST COMPUTER CONFIGURATION SUPPLIED BY THE NETWORK INFORMATION CENTER**

**NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES**

IMP TIP PLUF SATE



**ARPANET PROTOCOL HANDBOOK**  
**NIC 7104, Rev. Jan. 1978**

**ABSTRACT**

The ARPANET Protocol Handbook is a collection of documents that describe the protocols currently in use on the ARPANET Computer Network as of January 1978. Protocols are the rules of communication between processes on the network. The protocols in use on the ARPANET form a tree structure. The basic protocol is the IMP-to-Host Protocol. Directly under that is the Host-to-Host Protocol. Spreading out beneath, but still closely related, are the process level protocols: TELNET, File Transfer, Remote Job Entry, and Graphics. Interspersed along the way are a few small protocols such as the Initial Connection Protocol, and the definition of the standard character set.



TABLE-OF-CONTENTS  
ARPANET PROTOCOL HANDBOOK

PROTOCOL HANDBOOK TABLE-OF-CONTENTS

ABSTRACT .....	i
TABLE OF CONTENTS .....	iii
INTRODUCTION .....	1
IMP-HOST PROTOCOL .....	Not included
Interface Message Processor, Specifications for the Interconnection of a Host and an IMP (available from Bolt, Beranek, and Newman, Inc. as BBN Rept. 1822 or from NTIS as AD A019160)	
NETWORK CONTROL PROTOCOLS .....	5
Primary Host to Host Protocol [NCP]	
Host/Host Protocol for the ARPANET (NIC 8246) .....	5
TIP/TENEX Reliability Improvements (RFC 636, NIC 30490) .....	35
Transmission Control Protocol [TCP] (NIC 29618,) .....	41
Network Voice Protocol [NVP] (RFC 741, NIC 42444) .....	43
INITIAL CONNECTION PROTOCOLS .....	79
Official Initial Connection Protocol (NIC 7101) .....	79
Possible Deadlock (RFC 202, NIC 7155) .....	85
Official TELNET-Logger Initial Connection Protocol (NIC 7103) .....	87
TELNET PROTOCOLS .....	89
TELNET Protocol Specification (NIC 18639) .....	89
TELNET Option Specifications (NIC 18640) .....	99
Index of Telnet Option Numbers (NIC 29610) .....	101
TELNET Binary Transmission Option (NIC 15389) .....	103
TELNET Echo Option (NIC 15390) .....	107
TELNET Reconnection Option (NIC 15391) .....	113
TELNET Suppress Go Ahead Option (NIC 15392) .....	123

**TABLE-OF-CONTENTS**  
**ARPANET PROTOCOL HANDBOOK**

<b>TELNET Approximate Message Size Negotiation Option (NIC 15393) . . . . .</b>	<b>125</b>
<b>Revised TELNET Status Option (RFC 651, NIC 31154) . . . . .</b>	<b>129</b>
<b>TELNET Timing Mark Option (NIC 16238) . . . . .</b>	<b>133</b>
<b>TELNET Remote Controlled Transmission and Echoing Option (RFC 726, NIC 39237) . . . . .</b>	<b>137</b>
<b>TELNET Output Line Width Option (NIC 20196) . . . . .</b>	<b>153</b>
<b>TELNET Output Page Size Option (NIC 20197) . . . . .</b>	<b>157</b>
<b>TELNET Output Carriage-Return Disposition Option (RFC 652) . . . . .</b>	<b>161</b>
<b>TELNET Output Horizontal Tabstops Option (RFC 653) . . . . .</b>	<b>165</b>
<b>TELNET Output Horizontal Tab Disposition Option (RFC 654) . . . . .</b>	<b>169</b>
<b>TELNET Output Formfeed Disposition Option (RFC 655) . . . . .</b>	<b>173</b>
<b>TELNET Output Vertical Tabstops Option (RFC 656) . . . . .</b>	<b>177</b>
<b>TELNET Output Vertical Tab Disposition Option (RFC 657) . . . . .</b>	<b>181</b>
<b>TELNET Output Linefeed Disposition (RFC 658) . . . . .</b>	<b>185</b>
<b>TELNET Extended ASCII Option (RFC 698, NIC 32964) . . . . .</b>	<b>189</b>
<b>TELNET Logout Option (RFC 727, NIC 40025) . . . . .</b>	<b>193</b>
<b>TELNET Byte Macro Option (RFC 735, NIC 42083) . . . . .</b>	<b>197</b>
<b>TELNET Data Entry Terminal Option (RFC 732, NIC 41762) . . . . .</b>	<b>203</b>
<b>TELNET SUPDUP Option (RFC 736, NIC 42213) . . . . .</b>	<b>233</b>
<b>TELNET SUPDUP Protocol (RFC 734, NIC 41953) . . . . .</b>	<b>235</b>
<b>TELNET Extended-Options-List Option (NIC 16239) . . . . .</b>	<b>249</b>
<b>USA Standard Code for Information Interchange (NIC 11246) . . . . .</b>	<b>251</b>

**TABLE OF CONTENTS  
ARPANET PROTOCOL HANDBOOK**

<b>FILE TRANSFER PROTOCOL AND STANDARDS.....</b>	<b>265</b>
File Transfer Protocol (RFC 542, NIC 17759) .....	265
Revised FTP Reply Codes (RFC 640, NIC 30843) .....	299
FTP Extension: XSEN (RFC 737, NIC 42217).....	317
FTP Extension: XTP (RFC 683, NIC 32251).....	319
Document File Format Standards (RFC 678, NIC 31524).....	325
<b>MAIL PROTOCOL AND STANDARDS.....</b>	<b>333</b>
Mail Protocol (NIC 29588) .....	333
Standard for the Format of ARPA Network Text Messages (RFC 733, NIC 41592) ..	335
<b>REMOTE JOB ENTRY PROTOCOLS .....</b>	<b>379</b>
Remote Job Entry Protocol (RFC 407, NIC 12112).....	379
NETRJS Protocol (RFC 740, NIC 42423) .....	399
<b>NETWORK GRAPHICS PROTOCOLS.....</b>	<b>421</b>
A Network Graphics Protocol (NIC 24308).....	421
<b>MISCELLANEOUS PROTOCOLS .....</b>	<b>479</b>
Time Server Protocol (RFC 738, NIC 42218).....	479
Finger/Name Protocol (NIC 42758) .....	481
<b>NUMBER ASSIGNMENTS .....</b>	<b>489</b>
Assigned Numbers (RFC 739, NLS 42341) .....	489
<b>BIBLIOGRAPHY .....</b>	<b>501</b>
Description of NSW Protocols (NIC 29613) .....	501
General Bibliography.....	509

Numbers listed in the Table of Contents refer to the page numbers in parentheses. Center paging pertains only to the protocol in which it occurs. Center paging has been included so that page references within the text of a given protocol that refer to the original paging will still have meaning.



**CONTACTS**  
**ARPANET PROTOCOL HANDBOOK**

**CONTACTS**

**ARPANET MANAGEMENT, POLICY, AND SERVICE**

Questions regarding general policy or access to the ARPANET or unsatisfactory ARPANET service should be directed to:

Director, Defense Communications Agency (DCA)  
ATTN: ARPANET Management Branch, Maj. Raymond Czahor  
Code 535  
Washington, D. C. 20305

(202) 692-6175 or 692-6176 (Commercial)  
222-6175 or 222-6176 (Autovon)  
DCACODE535@ISI (Online mail)

**PROTOCOLS**

Technical questions regarding network protocols should be directed to:

Network Working Group Coordinator (NWG)  
ATTN: Jon Postel  
University of Southern California  
Information Science Institute  
4676 Admiralty Way  
Marina Del Rey, California 90291

(213) 822-1511  
POSTEL@USC-ISIB (Online mail)

**NETWORK OPERATION AND TRAFFIC**

Technical questions concerning network operation and usage should be directed to:

Network Control Center (NCC)  
ATTN: Paul Santos  
Bolt Beranek and Newman, Inc.  
Cambridge, Massachusetts 02138

(617) 491-1850 ext 603  
SANTOS@BBNE (Online mail)

**CONTACTS**  
**ARPANET PROTOCOL HANDBOOK**

**IMPS, TIPS, AND NETWORK LINE PROBLEMS**

Routine questions on IMP and TIP operation and network line problems should be directed to:

Network Control Center (NCC)  
ATTN: Jim Powers  
Bolt Beranek and Newman, Inc.  
Cambridge, Massachusetts 02138

(617) 661-0100  
POWERS@BBNE (Online mail)

**GENERAL NETWORK RESOURCE INFORMATION**

Questions regarding what or where resources are located on the network or who to contact should be directed to:

Network Information Center (NIC)  
ATTN: Elizabeth Feinler  
SRI International  
Menlo Park, California 94025

(415) 326-6200 ext 3695  
FEINLER@SRI-KL (Online mail)

**SPECIFIC HOST INFORMATION**

Questions regarding the use of a specific resource on a given host should be directed to the LIAISON for that host. A list of the Network Liaison is available from host SRI-KL (66 dec) as file, <NETINFO>LIAISON.TXT. This file may be FTPed to your local host using the 'anonymous' login convention.

## INTRODUCTION ARPANET PROTOCOL HANDBOOK

### BRIEF EXPLANATION OF THE PROTOCOL HANDBOOK

#### WHAT THE PROTOCOL HANDBOOK IS

The ARPANET Protocol Handbook (NIC 7104) is a document that describes specifications for the protocols in use on the ARPANET computer network. The Handbook includes all the current accepted protocols except the IMP-HOST Protocol (Interface Message Processor, Specifications for the Interconnection of a Host and an IMP, Rept. 1822, Bolt Beranek and Newman, Inc., Cambridge, Mass., Rev. Jan. 1976). Since this particular protocol is very long and is distributed to the Liaison of each host on the ARPANET, it has been omitted from this volume. The IMP-HOST Protocol may be obtained from the National Technical Information Service (NTIS), Springfield, Va. 22161 using order number, AD A019160.

#### HOW PROTOCOL INFORMATION IS OBTAINED

Information included in the Protocol Handbook is supplied by the Coordinator of the ARPANET Network Working Group to the Network Information Center at the request of the Defense Communications Agency. Dr. Jon Postel, University of Southern California, Information Sciences Institute, is the current Network Working Group Coordinator (POSTEL@ISIB).

#### ONLINE ACCESS TO ARPANET PROTOCOLS VIA THE ARPANET

Users of the ARPANET may access the protocols online by connecting to SRI-KL (Host 66, dec.) and then typing the word 'nic' followed by a <CR>. Follow the instructions given, and choose the menu item, 'Protocols'. The files listed may also be FTPed to your local host by using the 'anonymous' login convention.

#### REQUESTS FOR COMMENTS (RFCs)

Before a proposed protocol is accepted for use on the ARPANET it is discussed, reviewed, and often revised by members of the Network Working Group and other interested parties. This dialog is captured in a set of technical notes known as Requests for Comments or RFCs. ARPANET users may obtain the RFCs from the directory <NETINFO> at SRI-KL through FTP, using the 'anonymous' login convention.

New RFCs stay online a minimum of 21 days. After that time they may be archived. If you cannot locate a given RFC, send an online request to FEINLER@SRI-KL.

Members of the ARPANET who wish to be on distribution for online notices of new RFCs should contact Jon Postel (POSTEL@ISIB).

#### ORDERING THE HANDBOOK

The Protocol Handbook may be ordered by official ARPANET users from the

Network Information Center (NIC)  
ATTN: Elizabeth Feinler  
SRI International  
Menlo Park, California 94025

**INTRODUCTION**  
**ARPANET PROTOCOL HANDBOOK**

or by the general public from the

National Technical Information Service (NTIS)  
5285 Port Royal Road  
Springfield, Virginia 22161

NOTE: A reference copy will be sent to each Liaison and Network  
Sponsor on the ARPANET at the time the handbook is published.

**USE OF THE ARPANET**

The ARPANET is intended to be used solely for the conduct of, or in support of, official U.S. Government business. Use of the ARPANET must not be in violation of information privacy laws and is not intended to compete with existing commercial services.

**HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)  
Obsoletes NIC 7147**

**HOST-TO-HOST PROTOCOL  
for the  
ARPANET**

Prepared for the Network Working Group by

Alex McKenzie  
Bolt Beranek and Newman, Inc.  
Cambridge, Massachusetts 02138  
January 1972

Revised by

Jon Postel  
Information Sciences Institute  
Marina Del Rey, California 90291  
October 1977



**HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)**

**PREFACE**

This document specifies a protocol for use in communication between Host computers on the ARPANET. In particular, it provides for connection of independent processes in different Hosts, control of the flow of data over established connections, and several ancillary functions. Although basically self-contained, this document specifies only one of several ARPANET protocols; all protocol specifications are collected in the document ARPANET Protocol Handbook, NIC 7104.

This document supercedes NIC 7147 of the same title. Principal differences between the NIC 7147 and the January 1972 version of NIC 8246 were:

- . Prohibition of spontaneous RET, ERP, and RRP commands
- . A discussion of the problem of unanswered CLS commands
- . A discussion of the implications of queueing and not queueing RFCs
- . The strong recommendation that received ERR commands be logged; also some additional ERR specifications.

In this October 1977 revision of NIC 8246 the principal changes include:

- . Remarks on the relationship of the Host-to-Host Protocol to the revised Host/IMP protocol, especially the Host/IMP leader format
- . Explanation of the NCP's role in segmenting data for transmission
- . Extension of the ERR command specification

In addition to the above, several minor editorial changes have been made.

Although there are many individuals associated with the network who are knowledgeable about protocol issues, questions pertaining to network protocols should initially be directed toward one of the following:

Alex McKenzie  
Bolt Beranek and Newman, Inc.  
50 Moulton Street  
Cambridge, Massachusetts 02138  
(617) 491-1850 ext. 441 [McKenzie@BBNE]

Jon Postel  
University of Southern California  
Information Sciences Institute  
4676 Admiralty Way  
Marina del Rey, California 90274  
(213) 822-1511 [Postel@ISIB]

**Preceding page blank**

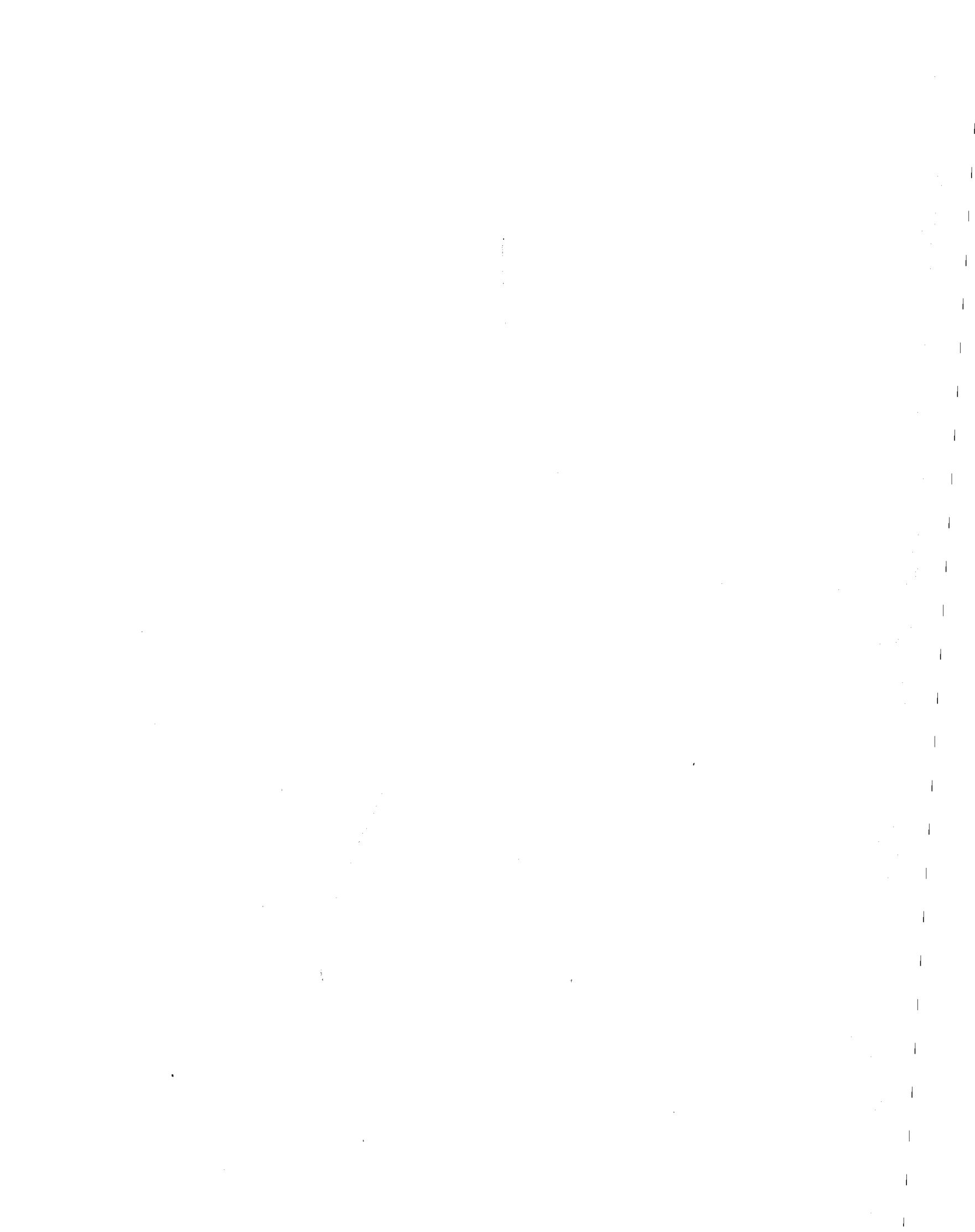


**HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)**

**TABLE OF CONTENTS**

I. INTRODUCTION .....	1
An overview of the multi-leveled protocol structure in the ARPANET.	
II. COMMUNICATION CONCEPTS .....	2
Definitions of terminology and a description of the overall strategy used in Host-to-Host communications.	
III. NCP FUNCTIONS .....	5
The meat of the document for the first-time reader. Host-to-Host "commands" are introduced with descriptions of conditions of their use, discussion of possible problems, and other background material.	
Connection Establishment .....	5
Connection Termination .....	7
Flow Control .....	8
Interrupts .....	10
Test Inquiry .....	11
Reinitialization .....	12
IV. DECLARATIVE SPECIFICATIONS.....	13
Details for the NCP implementer. A few additional "commands" are introduced, and those described in Section III are reviewed. Formats and code and link assignments are specified.	
Message Format .....	13
Link Assignment .....	14
Control Messages .....	15
Control Commands .....	15
Opcode Assignments .....	21
Control Command Summary .....	22

**Preceding page blank**



**HOST-TO-HOST PROTOCOL  
FOR THE  
ARPANET**

**INTRODUCTION**

The ARPANET provides a capability for geographically separated computers, called Hosts, to communicate with each other. The Host computers typically differ from one another in type, speed, word length, operating system, etc. Each Host computer is connected into the network through a local small computer called an Interface Message Processor (IMP). The complete network is formed by interconnecting these IMPs, all of which are virtually identical, through wideband communications lines supplied by the telephone company. Each IMP is programmed to store and forward messages to the neighboring IMPs in the network. During a typical operation, a Host passes a message to its local IMP; the first few bits (called the "leader") of this message include the "network address" of a destination Host. The message is passed from IMP to IMP through the Network until it finally arrives at the destination IMP, which in turn passes it along to the destination Host.

Specifications for the physical and logical message transfer between a Host and its local IMP are contained in Bolt Beranek and Newman (BBN) Report No. 1822 (AD A019160), last updated January 1976. These specifications are generally called the first level protocol or Host/IMP Protocol. This protocol is not by itself, however, sufficient to specify meaningful communication between processes running in two dissimilar Hosts. Rather, the processes must have some agreement as to the method of initiating communication, the interpretation of transmitted data, and so forth. Although it would be possible for such agreements to be reached by each pair of Hosts (or processes) interested in communication, a more general arrangement is desirable in order to minimize the amount of implementation necessary for Network-wide communication. Accordingly, the Host organizations formed a Network Working Group (NWG) to facilitate an exchange of ideas and to formulate additional specifications for Host-to-Host communications.

The NWG has adopted a "layered" approach to the specification of communication protocols. The inner layer is the Host/IMP protocol. The next layer specifies methods of establishing communications paths, managing buffer space at each end of a communication path, and providing a method of "interrupting" a communication path. This protocol, which will be used by all higher-level protocols, is known as the second level protocol, or Host-to-Host protocol. (It is worth noting that, although the IMP sub-network provides a capability for message switching, the Host/Host protocol is based on the concept of line switching.) Examples of further layers of protocol currently developed or anticipated include:

1. An Initial Connection Protocol (ICP) which provides a convenient standard method for several processes to gain simultaneous access to some specific process (such as the "logger") at another Host. [This is not so much a level of protocol as it is a procedure or subroutine used by third level protocols to establish communication.]
2. A Telecommunication Network (TELNET) protocol which provides for the "mapping" of an arbitrary Keyboard-printer terminal into a network Virtual Terminal (NVT), to facilitate communication between a terminal user at one Host site and a terminal-serving process at some other site which "expects" to be connected to a

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

(local) terminal logically different from the (remote) terminal actually in use. The TELNET protocol specifies use of the ICP to establish the communication path between the terminal user and the terminal-service process.

3. A File Transfer protocol to specify methods for reading, writing, and updating files stored at a remote Host. The File Transfer protocol includes specifications for the transmission of data in accordance with a set of data transfer formats.
4. A Remote Job Service (RJS) protocol to specify methods for submitting input to, obtaining output from, and exercising control over Hosts which provide batch processing facilities.
5. A Graphics protocol to specify the means for exchanging graphics display information.

The remainder of this document describes and specifies the Host-to-Host, or second level, protocol as formulated by the Network Working Group.

**COMMUNICATION CONCEPTS**

The IMP sub-network imposes a number of physical restrictions on communications between Hosts; these restrictions are presented in BBN Report Number 1822. In particular, the concepts of leaders, messages, padding, links, and message types are of interest to the design of Host-to-Host protocol. The following discussion assumes that the reader is familiar with these concepts.

Time out for a little history. When the Host-to-Host Protocol was first specified the Host/IMP interface was somewhat different than it is now. The major difference is that the Leader was 32 bits and is now 96 bits. Another change is that the 8 bit Leader field called "link", has been replaced by a 12 bit field called "message id". In the following when a link is mentioned what is intended is an 8 bit quantity, and if a link value is to be inserted into or read from a Host/IMP Leader, it is to be the high-order 8 bits of the message-id field (the low order 4 bits should be zero).

Although there is little uniformity among the Hosts in either hardware or operating systems, the notion of multiprogramming dominates most of the systems. These Hosts can each concurrently support several users, with each user running one or more processes. Many of these processes may want to use the network concurrently, and thus a fundamental requirement of the Host-to-Host protocol is to provide for process-to-process communication over the network. Since the first level protocol only takes cognizance of Hosts, and since the several processes in execution within a Host are usually independent, it is necessary for the second level protocol to provide a richer addressing structure.

Another factor which influenced the Host-to-Host protocol design was the expectation that typical process-to-process communication would be based, not on a solitary message, but rather upon a sequence of messages. One example is the sending of a large body of information, such as a data base, from one process to another. Another example is an interactive conversation between two processes, with many exchanges.

These considerations led to the introduction of the notions of connections, a Network Control program, a "control link", "control commands", connection byte size, message headers, and sockets.

HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)

A connection is an extension of a link. A connection couples two processes so that output from one process is input to the other. Connections are defined to be simplex (i.e., unidirectional), so two connections are necessary if a pair of processes are to converse in both directions.

Processes within a Host are envisioned as communicating with the rest of the network through a Network Control Program (NCP), resident in that Host, which implements the second level protocol. The primary function of the NCP is to establish connections, break connections, and control data flow over the connections. We will describe the NCP as though it were part of the operating system of a Host supporting multiprogramming, although the actual method of implementing the NCP may be different in some Hosts.

In order to accomplish its tasks, the NCP of one Host must communicate with the NCPs of other Hosts. To this end, a particular link between each pair of Hosts has been designated as the CONTROL LINK. Messages transmitted over the control link are called CONTROL MESSAGES, (NOTE: In BBN Report Number 1822, messages of non-zero type are called control messages, and are used to control the flow of information between a Host and its IMP. In this document, the term "control message" is used for a message of type zero transmitted over the control link. The IMPs take no special notice of these messages) and must always be interpreted by an NCP as a sequence of one or more CONTROL COMMANDS. For example, one kind of control command is used to initiate a connection, while another kind carries notification that a connection has been terminated.

The concept of a message, as used above, is an artifact of the IMP sub-network; network message boundaries may have little intrinsic meaning to communication processes. Accordingly, it has been decided that the NCP (rather than each sending process) should be responsible for segmenting interprocess communication into network messages. Therefore, it is a principle of the second level protocol that no significance may be inferred from message boundaries by a receiving process. *The only exception to this principle is in control messages, each of which must contain an integral number of control commands.*

Please note that the protocol provides mechanisms for flow control which may require that data be transmitted in blocks whose size can vary over time and which at times may be much smaller than the maximum message size allowed by the subnet. This is a further reason for the NCP to be responsible for segmenting the data into transmission units, rather than the sending process.

Since message boundaries are selected by the transmitting NCP, the receiving NCP must be prepared to concatenate successive messages from the network into a single (or differently divided) transmission for delivery to the receiving process. The fact that Hosts have different word sizes means that a message from the network might end in the middle of a word at the receiving end, and thus the concatenation of the next message might require the receiving Host to carry out extensive bit-shifting. Because bit-shifting is typically very costly in terms of computer processing time, the protocol includes the notions of connection byte size and message headers.

As part of the process of establishing a connection, the processes involved must agree on a CONNECTION BYTE SIZE. Each message sent over the connection must then contain an integral number of bytes of this size. Thus the pair of processes involved in a connection can choose a mutually convenient byte size, for example, the least common multiple of their Host word lengths. It is important to note that the ability to choose a byte size MUST be available

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

to the processes involved in the connection; an NCP is prohibited from imposing an arbitrary byte size on any process running in its own Host. In particular, an outer layer of protocol may specify a byte size to be used by that protocol. If some NCP is unable to handle that byte size, then the outer layer of protocol will not be implementable on that Host.

The IMP sub-network requires that the first few bits of each message (called the leader) contain addressing information, including destination Host address and link number. The second level protocol extends the required information at the beginning of each message by 40 bits; the Leader and the additional 40 bits together are called the MESSAGE HEADER.

The old style leader was 32 bits, thus the total length of the message header was 72 bits. A length of 72 bits was chosen since most Hosts either can work conveniently with 8-bit units of data or have word lengths of 18 or 36 bits; 72 is the least common multiple of these lengths. Thus, the length chosen for the message header was intended to reduce bit-shifting problems for many Hosts. The new style leader is 96 bits, thus the total message header is 136 bits. This is 17 octets, and not particularly convenient for any Host.

In addition to the leader, the message header includes a field giving the byte size used in the message, a field giving the number of bytes in the message, and "filler" fields. The format of the message header is fully described in Section IV.

Another major concern of the second level protocol is a method for referencing processes in other Hosts. Each Host has some internal scheme for naming processes, but these various schemes are typically different and may even be incompatible. Since it is not practical to impose a common internal process naming scheme, a standard intermediate name space is used, with a separate portion of the name space allocated to each Host. Each Host must have the ability to map internal process identifiers into its portion of this name space.

The elements of the name space are called SOCKETS. A socket forms one end of a connection, and a connection is fully specified by a pair of sockets. A socket is identified by a Host number and a 32-bit socket number. The same 32-bit number in different Hosts represents different sockets.

A socket is either a RECEIVE SOCKET or a SEND SOCKET, and is so marked by its low-order bit (0 = receive; 1 = send). This property is called the socket's GENDER. The sockets at either end of a connection must be of opposite gender. Except for the gender, second level protocol places no constraints on the assignment of socket numbers within a Host.

HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)

## NCP FUNCTIONS

The functions of the NCP are to establish connections, terminate connections, control flow, transmit interrupts, and respond to test inquiries. These functions are explained in this section, and control commands are introduced as needed. In Section IV the formats of all control commands are presented together.

### CONNECTION ESTABLISHMENT

The commands used to establish a connection are STR (sender-to-receiver) and RTS (receiver-to-sender).

8*	32	32	8
<hr/>			
! STR !	send socket	!	receive socket ! size !
<hr/>			
8	32	32	8
<hr/>			
! RTS !	receive socket	!	send socket ! link !
<hr/>			

NOTE: The number shown above each control command field is the length of that field in bits.

The STR command is sent from a prospective sender to a prospective receiver, and the RTS from a prospective receiver to a prospective sender. The send socket field names a socket local to the prospective sender; the receive socket field names a socket local to the prospective receiver. In the STR command, the "size" field contains an unsigned binary number (in the range 1 to 255; zero is prohibited) specifying the byte size to be used for all messages over the connection. In the RTS command, the "link" field specifies a link number; all messages over the connection must be sent over the link specified by this number. These two commands are referred to as requests-for-connection (RFCs). An STR and an RTS match if the receive socket fields match and the send socket fields match. A connection is established when a matching pair of RFCs have been exchanged.  
*Hosts are prohibited from establishing more than one connection to any local socket.*

With respect to a particular connection, the Host containing the send socket is called the SENDING HOST and the Host containing the receive socket is called the RECEIVING HOST. A Host may connect one of its receive sockets to one of its send sockets, thus becoming both the sending Host and the receiving Host for that connection. These terms apply only to data flow; control messages will, in general, be transmitted in both directions.

A Host sends an RFC either to request a connection, or to accept a foreign Host's request. Since RFC commands are used both for requesting and for accepting the establishment of a connection, it is possible for either of two cooperating processes to initiate connection establishment. As a consequence, a family of processes may be created with connection-initiating actions built-in, and the processes within this family may be started up (in different Hosts) in arbitrary order provided that appropriate queueing is performed by the Hosts involved (see below).

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

*There is no prescribed lifetime for an RFC.* A Host is permitted to queue incoming RFCs and withhold a response for an arbitrarily long time, or, alternatively, to reject requests (see Connection Termination below) immediately if it does not have a matching RFC outstanding. It may be reasonable, for example, for an NCP to queue an RFC that refers to some currently unused socket until a local process takes control of that socket number and tells the NCP to accept or reject the request. Of course, the Host which sent the RFC may be unwilling to wait for an arbitrarily long time, so it may abort the request. On the other hand, some NCP implementations may not include any space for queueing RFCs, and thus can be expected to reject RFCs unless the RFC sequence was initiated locally.

#### QUEUEING CONSIDERATIONS

The decision to queue, or not queue, incoming RFCs has important implications which NCP implementers must not ignore. Each RFC which is queued, of course, requires a small amount of memory in the Host doing the queueing. If each incoming RFC is queued until a local process seizes the local socket and accepts (or rejects) the RFC, but no local process ever seizes the socket, the RFC must be queued "forever." Theoretically this could occur infinitely many times (there is no reason not to queue several RFCs for a single local socket, letting the local process decide which, if any, to accept) thus requiring infinite storage for the RFC queue. On the other hand, if no queueing is performed the cooperating processes described above will be able to establish a desired connection only by accident (when they are started up such that one issues its RFC while the RFC of the other is in transit in the network - clearly an unlikely occurrence).

Perhaps the most reasonable solution to the problems posed above is for EACH NCP to give processes running in its own Host two options for attempting to initiate connections. The first option would allow a process to cause an RFC to be sent to a specified remote socket; with the NCP notifying the process as to whether the RFC were accepted or rejected by the remote Host. The second option would allow a process to tell ITS OWN NCP to "listen" for an RFC to a specified local socket from some remote socket (the process might also specify the particular remote socket and/or Host it wishes to communicate with) and to accept the RFC (i.e., return a matching RFC) if and when it arrives. Note that this also involves queueing (of "listen" requests), but it is internal queueing which is susceptible to reasonable management by the local Host. If this implementation were available, one of two cooperating processes could "listen" while the other process caused a series of RFCs to be sent to the "listening" socket until one was accepted. Thus, no queueing of incoming RFCs would be required, although it would do no harm.

*It is the intent of the protocol that each NCP should provide either the "listen" option described above or a SUBSTANTIAL queueing facility.* This is not, however, an absolute requirement of the protocol.

## CONNECTION TERMINATION

The command used to terminate a connection is CLS (close).

8	32	32
! CLS !	my socket	! your socket !

The "my socket" field contains the socket local to the sender of the CLS command. The "your socket" field contains the socket local to the receiver of the CLS command. *Each side must send and receive a CLS command before connection termination is completed and the sockets are free to participate in other connections.*

It is not necessary for a connection to be established (i.e., for BOTH RFCs to be exchanged) before connection termination begins. For example, if a Host wishes to refuse a request for connection, it sends back a CLS instead of a matching RFC. The refusing Host then waits for the initiating Host to acknowledge the refusal by returning a CLS. Similarly, if a Host wishes to abort its outstanding request for a connection, it sends a CLS command. The foreign Host is obliged to acknowledge the CLS with its own CLS. Note that even though the connection was never established, CLS commands must be EXCHANGED before the sockets are free for other use.

After a connection is established, CLS commands sent by the receiver and sender have slightly different effects. CLS commands sent by the sender indicate that no more messages will be sent over the connection. *This command must not be sent if there is a message in transit over the connection.* A CLS command sent by the receiver acts as a demand on the sender to terminate transmission. However, since there is a delay in getting the CLS command to the sender, the receiver must expect more input.

A Host should "quickly" acknowledge an incoming CLS so the foreign Host can purge its tables. However, *there is no prescribed time period in which a CLS must be acknowledged.*

Because the CLS command is used both to initiate closing, aborting and refusing a connection, and to acknowledge closing, aborting and refusing a connection, race conditions can occur. However, they do not lead to ambiguous or erroneous results, as illustrated in the following examples:

EXAMPLE 1: Suppose that Host A sends B a request for connection, and then A sends a CLS to Host B because it is tired of waiting for a reply. However, just when A sends its CLS to B, B sends a CLS to A to refuse the connection. A will "believe" B is acknowledging the abort, and B will "believe" A is acknowledging its refusal, but the outcome will be correct.

EXAMPLE 2: Suppose that Host A sends Host B an RFC followed by a CLS as in example 1. In this case, however, B sends a matching RFC to A just when A sends its CLS. Host A may "believe" that the RFC is an attempt (on the part of B) to establish a new connection or may understand the race condition; in either case it can discard

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

the RFC since its socket is not yet free. Host B will "believe" that the CLS is breaking an ESTABLISHED connection, but the outcome is correct since a matching CLS is the required response, and both A and B will then terminate the connection.

Every NCP implementation is faced with the problem of what to do if a matching CLS is not returned "quickly" by a foreign Host (i.e., if the foreign Host appears to be violating protocol in this respect). One naive answer is to hold the connection in a partially closed state "forever" waiting for a matching CLS. There are two difficulties with this solution. First, the socket involved may be a "scarce resource" such as the "logger" socket specified by an Initial Connection Protocol (see NIC 7101) which the local Host cannot afford to tie up indefinitely. Second, a partially broken (or malicious) process in a foreign Host may send an unending stream of RFCs which the local Host wishes to refuse by sending CLS commands and waiting for a match. This could, in worst cases, require 2 to the 32nd factorial socket pairs to be stored before duplicates began to appear. Clearly, no host is prepared to store (or search) this much information.

A second possibility sometimes suggested is for the Host which is waiting for matching CLS commands (Host A) to send a RST (see page 17) to the offending Host (Host B), thus allowing all tables to be reinitialized at both ends. This would be rather unsatisfactory to any user at Host A who happened to be performing useful work on Host B via network connections, since these connections would also be broken by the RST.

Most implementers, recognizing these problems have adopted some unofficial timeout period after which they "forget" a connection even if a matching CLS has not been received. The danger with such an arrangement is that if a second connection between the same pair of sockets is later established, and a CLS finally arrives for the first connection, the second connection is likely to be closed. This situation can only arise, however, if one Host violates protocol in TWO ways; first by failing to respond quickly to an incoming CLS, and second by permitting establishment of a connection involving a socket which it believes is already in use. It has been suggested that the network adopt some standard timeout period, but the NWG has been unable to arrive at a period which is both short enough to be useful and long enough to be acceptable to every Host.

Timeout periods in current use seem to range between approximately one minute and approximately five minutes. *It must be emphasized that all timeout periods, although they are relatively common, reasonably safe, and quite useful, are in violation of the protocol since their use can lead to connection ambiguities.*

#### FLOW CONTROL

After a connection is established, the sending Host sends messages over the agreed-upon link to the receiving Host. The receiving NCP accepts messages from its IMP and queues them for its various processes. Since it may happen that the messages arrive faster than they can be processed, some mechanism is required which permits the receiving Host to quench the flow from the sending Host.

The flow control mechanism requires the receiving Host to allocate buffer space for each connection and to notify the sending Host of how much space is available. The sending Host keeps track of how much room is available and never sends more data than it believes the receiving Host can accept.

HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)

To implement this mechanism, the sending Host keeps two counters associated with each connection, a MESSAGE COUNTER and a BIT COUNTER. Each counter is initialized to zero when the connection is established and is increased by allocate (ALL) control commands sent from the receiving Host as described below. When sending a message, the NCP of the sending Host subtracts one from the message counter and the TEXT LENGTH (defined below) from the bit counter. The sender is prohibited from sending if either counter would be decremented below zero. The sending Host may also return (RET) command upon receiving a give-back (GVB) command from the receiving Host (see below).

The TEXT LENGTH of a message is defined as the product of the connection byte size and the byte count for the message; both of these quantities appear in the message header. Messages with a zero byte count, hence a zero text length, are specifically permitted. Messages with zero text length do not use bit space allocation, but do use message space allocation. The flow control mechanisms do not pertain to the control link, since connections are never explicitly established over this link.

The control command used to increase the sender's bit counter and message counter is ALL (allocate).

8	8	16	32
-----			
! ALL	! link	! msg space	! bit space
-----			

This command is sent only from the receiving Host to the sending Host, and is legal only when a connection using the link number appearing in the "link" field is established. The "msg space" field and the "bit space" field are defined to be unsigned binary integers specifying the amounts by which the sender's message counter and bit counter (respectively) are to be incremented. The receiver is prohibited from incrementing the sender's message counter above 2 to the 16th minus 1, or the sender's bit counter above 2 to the 32nd minus 1. In general, this rule will require the receiver to maintain counters which are incremented and decremented according to the same rules as the sender's counters.

The receiving Host may request that the sending Host return all or part of its current allocation. The control command for this request is GVB (give-back).

8	8	8	8
-----			
! GVB	! link	! f(m)	! f(b)
-----			

This command is sent only from the receiving Host to the sending Host, and is legal only when a connection using the link number in the "link" field is established. The fields f(m) and f(b) are defined as the fraction (in 128ths) of the current message space allocation and bit space allocation (respectively) to be RETURNED. If either of the fractions is equal to or greater than one, ALL of the corresponding allocation must be returned. Fractions are used since, with messages in transit, the sender and receiver may not agree on the actual allocation at every point in time.

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

Upon receiving a GVB command, the sending Host must return AT LEAST (NOTE: In particular, fractional returns must be rounded up, not truncated) the requested portions of the message and bit space allocations. (A sending Host is prohibited from spontaneously returning portions of the message and bit space allocations.)

The control command for performing this return function is RET (return).

8	8	16	32
-----			
! RET	! link	! msg space	! bit space
-----			

This command is sent only from the sending Host to the receiving Host, and is legal only when a connection using the link number in the "link" field is established and a GVB command has been received from the receiving Host. The "msg space" field and the "bit space" field are defined as unsigned binary integers specifying the amounts by which the sender's message counter and bit counter (respectively) have been decremented due to the RET activity (i.e., the amounts of message and bit space allocation being returned). NCPs are obliged to answer a GVB with a RET "quickly"; however, there is no prescribed time period in which the answering RET must be sent.

Some Hosts will allocate only as much space as they can guarantee for each link. These Hosts will tend to use the GVB command only to reclaim space which is being filled very slowly or not at all. Other Hosts will allocate more space than they have, so that they may use their space more efficiently. Such a Host will then need to use the GVB command when the input over a particular link comes faster than it is being processed.

This flow control mechanism will result in variations in the size of blocks of data allowed to be transmitted. For most implementations the NCP will buffer the sending process's data and transmit data in blocks of a size equal to the maximum of the three values: the maximum subnet message size, the data available in the buffer to send, and the allocation. It may frequently be the case that the allocation is much smaller than the other two quantities.

#### INTERRUPTS

The second level protocol has included a mechanism by which the transmission over a connection may be "interrupted." The meaning of the "interrupt" is not defined at this level, but is made available for use by outer layers of protocol.

The interrupt command sent from the receiving Host to the sending Host is INR (interrupt-by-receiver).

8	8
-----	
! INR	! link
-----	

HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)

The interrupt command sent from the sending Host to the receiving Host is INS (interrupt-by-sender).

8        8  
-----  
! INS ! link !  
-----

The INR and INS commands are legal only when a connection using the link number in the "link" field is established.

TEST INQUIRY

It may sometimes be useful for one Host to determine if some other Host is capable of carrying on network conversations. The control command to be used for this purpose is ECO (echo).

8        8  
-----  
! ECO ! data !  
-----

The "data" field may contain any bit configuration chosen by the Host sending the ECO. Upon receiving an ECO command an NCP must respond by returning the data to the sender in an ERP (echo-reply) command.

8        8  
-----  
! ERP ! data !  
-----

A Host should "quickly" respond (with an ERP command) to an incoming ECO command. However, there is no prescribed time period, after the receipt of an ECO, in which the ERP must be returned. A Host is prohibited from sending an ERP when no ECO has been received, or from sending an ECO to a Host while a previous ECO to that Host remains "unanswered." Any of the following constitute an "answer" to an ECO: information from the local IMP that the ECO was discarded by the network (e.g., IMP/Host message type 7 - Destination Dead), ERP, RST, or RRP (see below).

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

**REINITIALIZATION**

Occasionally, due to lost control messages, system "crashes", NCP errors, or other factors, communication between two NCPs will be disrupted. One possible effect of any such disruption might be that neither of the involved NCPs could be sure that its stored information regarding connections with the other Host matched the information stored by the NCP of the other Host. In this situation, an NCP may wish to reinitialize its tables and request that the other Host do likewise; for this purpose the protocol provides the pair of control commands RST (reset) and RRP (reset-reply).

8

-----  
! RST !  
-----

8

-----  
! RRP !  
-----

The RST command is to be interpreted by the Host receiving it as a signal to purge its NCP tables of any entries which arose from communication with the Host to which sent the RST. The Host sending the RST should likewise purge its NCP tables of any entries which arise from communication with the Host to which the RST was sent. The Host receiving the RST should acknowledge receipt by returning an RRP. *Once the first Host has sent an RST to the second Host, the first Host is not obliged to communicate with the second Host (except for responding to RST) until the second Host returns an RRP.* In fact, to avoid synchronization errors, the first Host SHOULD NOT communicate with the second until the RST is answered. Of course, if the IMP subnetwork returns a "Destination Dead" (type 7) message in response to the control message containing the RST, an RRP should not be expected. If both NCPs decide to send RSTs at approximately the same time, then each Host will receive an RST and each must answer with an RRP, even though its own RST has not yet been answered.

It once seemed practical for some Hosts to "broadcast" RSTs to the entire network when they "came up." One method of accomplishing this would be to send an RST command to each of the possible Host addresses; the IMP subnetwork would return a "Destination Dead" (type 7) message for each non-existent Host, as well as for each Host actually "dead." *However, no Host is ever obliged to transmit an RST command.*

With the old style leader there were 256 possible host addresses, with the new leader there are 16,777,216 possible host addresses.

Hosts are prohibited from sending an RRP when no RST has been received. Further, Hosts may send only one RST in a single control message and should wait a "reasonable time" before sending another RST to the same Host. Under these conditions, a single RRP constitutes an "answer" to ALL RSTs sent to that Host, and any other RRP arriving from that Host should be discarded.

DECLARATIVE SPECIFICATIONS

MESSAGE FORMAT

All Host-to-Host messages (i.e., messages of type zero) shall have a header 136 bits long consisting of the following fields (see Figure 1):

- |              |   |
|--------------|---|
| Bits 1-96    | Leader – The contents of this field must be constructed according to the specifications contained in BBN Report Number 1822.  |
| Bits 97-104  | Field M1 – Must be zero.  |
| Bits 105-112 | Field S – Connection byte size. This size must be identical to the byte size on the STR used in establishing the connection. If this message is being transmitted over the control link the connection byte size must be 8. |
| Bits 113-128 | Field C – Byte Count. This field specifies the number of bytes in the text portion of the message. A zero value in the C field is explicitly permitted.   |
| Bits 129-136 | Field M2 – Must be zero.  |

Following the header, the message shall consist of a text field of C bytes, where each byte is S bits in length. Following the text there will be field M3 followed by padding. The M3 field is zero or more bits long and must be all zero; this field may be used to fill out a message to a word boundary.

NOTE: The Leader is 96 bits (or 3 words in this representation). Some hosts may continue to use the old style leader of 32 bits.

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

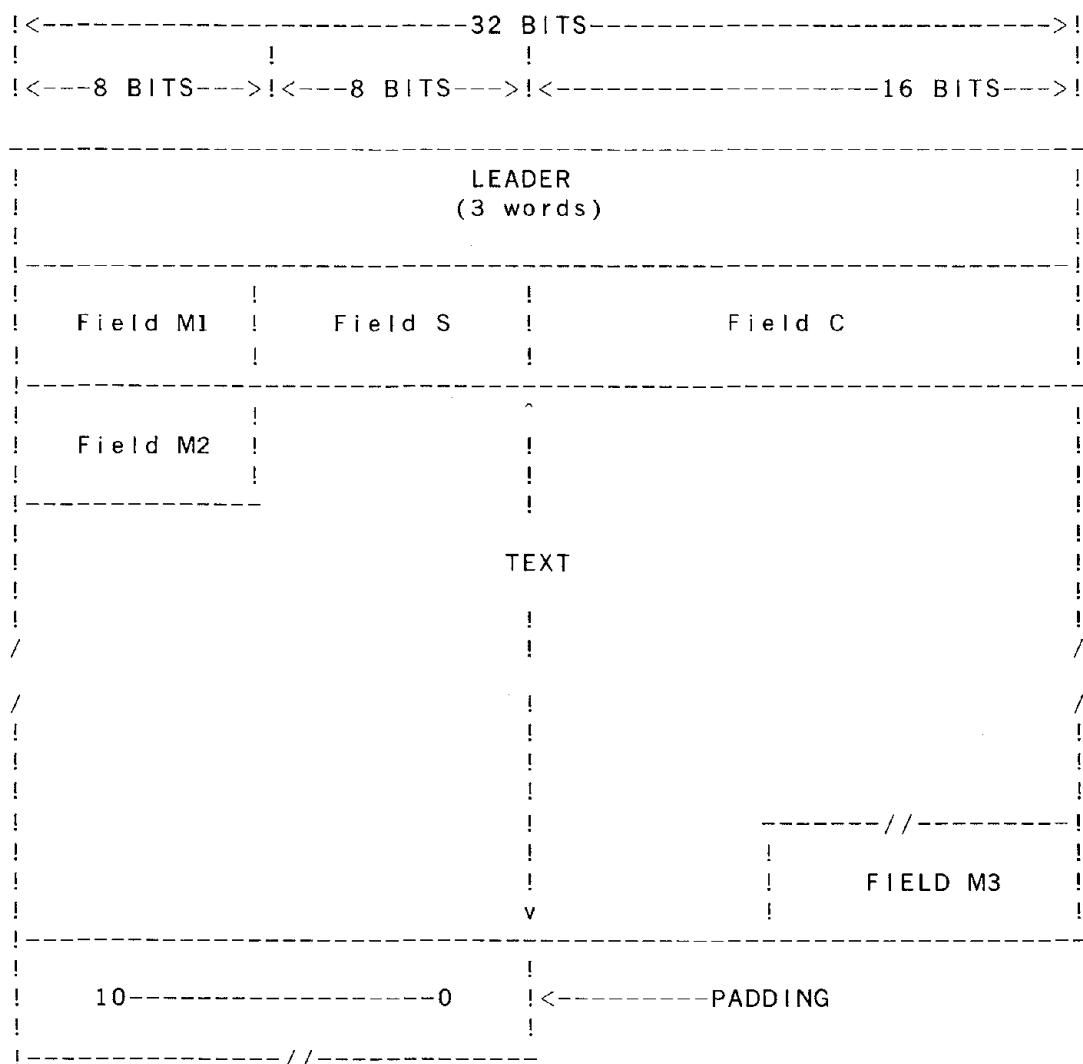


Figure 1.

The message header must, among other things, enable the NCP at the receiving Host to identify correctly the connection over which the message was sent. Given a set of messages from Host A to Host B, the only field in the header under the control of the NCP at Host B is the link number (assigned via the RTS control command). Therefore, each NCP must insure that, at a given point in time, for each connection for which it is the receiver, a unique link is assigned. Recall that the link is specified by the sender's address and the link number; thus a unique link number must be assigned to each connection to a given Host.

## LINK ASSIGNMENT

The complete list of link assignments is found in another section of the Protocol Handbook. The links assigned for use with this protocol are:

Decimal	Octal	Use
0	0	Control Messages
2-71	2-107	Regular Messages

## CONTROL MESSAGES

Messages sent over the control link have the same format as other Host-to-Host messages. The connection byte size (Field S in the message header) must be 8. Control messages may not contain more than 120 bytes of text; thus the value of the byte count (Field C in the message header) must be less than or equal to 120.

Control messages must contain an integral number of control commands. A single control command may not be split into parts which are transmitted in different control messages.

## CONTROL COMMANDS

Each control command begins with an 8-bit opcode. These opcodes have values of 0, 1, ... to permit table lookup upon receipt. Private experimental protocols should be tested using opcodes of 255, 254, .... Most of the control commands are more fully explained in Section III.

NOP - No operation

8  
-----  
! NOP !  
-----

The NOP command may be sent at anytime and should be discarded by the receiver. It may be useful for formatting control messages.

RST - Reset

8  
-----  
! RST !  
-----

The RST command is used by one Host to inform another that all information regarding previously existing connections, including partially terminated connections, between the two Hosts should be purged from the NCP tables of the Host receiving the RST. Except for responding to RSTs, the Host which sent the RST is not obliged to communicate further with the other Host until an RRP is received in response.

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

RRP - Reset reply

8

-----  
! RRP !  
-----

The RRP command must be sent in reply to an RST command.

RTS - Request connection, receiver to sender

8

32

32

8

-----  
! RTS ! receive socket ! send socket ! link !  
-----

The RTS command is used to establish a connection and is sent from the Host containing the receive socket to the Host containing the send socket. The link number for message transmission over the connection is assigned with this command; the "link" field must be between 2 and 71, inclusive.

STR - Request connection, sender to receiver

8

32

32

8

-----  
! STR ! send socket ! receive socket ! size !  
-----

The STR command is used to establish a connection and is sent from the Host containing the send socket to the Host containing the receive socket. The connection byte size is assigned with this command; the size must be between 1 and 255, inclusive.

CLS - Close

8

32

32

-----  
! CLS ! my socket ! your socket !  
-----

The CLS command is used to terminate a connection. A connection need not be completely established before a CLS is sent.

HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)

ALL - Allocate

8	8	16	32
-----			
! ALL	! link	! msg space	! bit space !
-----			

The ALL command is sent from a receiving Host to a sending Host to increase the sending Host's space counters. This command may be sent only while the connection is established. The receiving Host is prohibited from incrementing the sending Host's message counter above (2 to the 16th minus 1) or bit counter above (2 to the 32nd minus 1).

GVB - Give back

8	8	8	8
-----			
! GVB	! link	! f(m)	! f(b) !
-----			
		! -----bit fraction	
		!-----message fraction	
-----			

The GVB command is sent from a receiving Host to a sending Host to request that the sending Host return all or part of its Message space and/or bit space allocations. The "fractions" specify what portion (in 128ths) of each allocation must be returned. This command may be sent only while the connection is established.

RET - Return

8	8	16	32
-----			
! RET	! link	! msg space	! bit space !
-----			

The RET command is sent from the sending Host to the receiving Host to return all or a part of its message space and/or bit space allocations in response to a GVB command. This command may be sent only while the connection is established.

INR - Interrupt by receiver

8	8
-----	
! INR	! link !
-----	

The INR command is sent from the receiving Host to the sending Host when the receiving process wants to interrupt the sending process. This command may be sent only while the connection is established.

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

INS - Interrupt by sender

```
8      8  
-----  
! INS ! link !  
-----
```

The INS command is sent from the sending Host to the receiving Host when the sending process wants to interrupt the receiving process. This command may be sent only while the connection is established.

ECO - Echo request

```
8      8  
-----  
! ECO ! data !  
-----
```

The ECO command is used only for test purposes. The data field may be any bit configuration convenient to the Host sending the ECO command.

ERP - Echo reply

```
8      8  
-----  
! ERP ! data !  
-----
```

The ERP command must be sent in reply to an ECO command. The data field must be identical to the data field in the incoming ECO command.

ERR - Error detected

```
8      8          80  
-----/ /-----  
! ERR ! code !       data           !  
-----/ /-----
```

The ERR command MAY be sent whenever a second level protocol error is detected in the input from another Host. In the case that the error condition has a predefined error code, the "code" field specifies the specific error, and the data field gives parameters. For other errors the code field is zero and the data field is idiosyncratic to the sender. Implementers of Network Control Programs are expected to publish timely information on their ERR commands.

The usefulness of the ERR command is compromised if it is merely discarded by the receiver. Thus, sites are urged to record incoming ERRs if possible, and to investigate their cause in conjunction with the sending site. The following codes are defined. Additional codes may be defined later.

a. Undefined (Error code = 0)

The "data" field is idiosyncratic to the sender.

b. Illegal opcode (Error code = 1)

An illegal opcode was detected in a control message. The "data" field contains the ten bytes of the control message beginning with the byte containing the illegal opcode. If the remainder of the control message contains less than ten bytes, fill will be necessary; the value of the fill is zeros.

c. Short parameter space (Error code = 2)

The end of a control message was encountered before all the required parameters of the control command being decoded were found. The "data" field contains the command in error; the value of any fill necessary is zeros.

d. Bad parameters (Error code = 3)

Erroneous parameters were found in a control command. For example, two receive or two send sockets in an STR, RTS, or CLS; a link number outside the range 2 to 71 (inclusive); an ALL containing a space allocation too large. The "data" field contains the command in error; The value of any fill necessary is zeros.

e. Request on a non-existent socket (Error code = 4)

A request other than STR or RTS was made for a socket (or link) for which no RFC has been transmitted in either direction. This code is meant to indicate to the NCP receiving it that functions are being performed out of order. The "data" field contains the command in error; the value of any fill necessary is zeros.

f. Request on a non-open socket (Error code = 5)

A control command other than STR or RTS or CLS refers to a socket (perhaps referenced by link number) that is not party to a fully established connection. The socket's inappropriate state could either be that only one RFC has been sent (not yet open) or that only one CLS has been sent (not yet closed). The "data" field contains the command in error; the value of any fill necessary is zeros.

g. Message on an unknown link (Error code = 6)

A message was received over a link which is not currently being used for any connection. The contents of the "data" field are the message header followed by the first eight bits of text (if any) or zeros.

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

h. Invalid host header (Error code = 7)

A message was received either over the control link or a valid user link that had a host header with invalid format. Examples of when this subtype would be appropriate are the following: the M1 or M2 fields were non zero, the byte size was invalid (not 8 for control link, zero for any link), or the declared length exceeded the actual length. The contents of the "data" field are the message header padded with eight bits of zeros.

**HOST-TO-HOST PROTOCOL**  
**NIC 8246 (Oct. 1977)**

**OPCODE ASSIGNMENT**

Opcodes are defined to be eight-bit unsigned binary numbers. The values assigned to opcodes are:

Name	Decimal	Octal
NOP	= 0	0
RTS	= 1	1
STR	= 2	2
CLS	= 3	3
ALL	= 4	4
GVB	= 5	5
RET	= 6	6
INR	= 7	7
INS	= 8	10
ECO	= 9	11
ERP	= 10	12
ERR	= 11	13
RST	= 12	14
RRP	= 13	15

**HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)**

**CONTROL COMMAND SUMMARY**

8

-----  
! NOP !  
-----

8

32

32

8

-----  
! RTS ! receive socket ! send socket ! link !  
-----

8

32

32

8

-----  
! STR ! send socket ! receive socket ! size !  
-----

8

32

32

-----  
! CLS ! my socket ! your socket !  
-----

8

8

16

32

-----  
! ALL ! link ! msg space ! bit space !  
-----

8

8

8

8

-----  
! GVB ! link ! f(m) ! f(b) !  
-----

8

8

16

32

-----  
! RET ! link ! msg space ! bit space !  
-----

8

8

-----  
! INR ! link !  
-----

8

8

-----  
! INS ! link !  
-----

8

8

-----  
! ECO ! data !  
-----

**HOST-TO-HOST PROTOCOL  
NIC 8246 (Oct. 1977)**

8 8  
-----  
! ERP ! data !  
-----

8 8 80  
-----/ /-----  
! ERR ! code ! data !  
-----/ /-----

8  
-----  
! RST !  
-----

8  
-----  
! RRP !  
-----



**TIP/TENEX RELIABILITY IMPROVEMENTS  
RFC 636, NIC 30490 (June 10, 1974)**

**TIP/TENEX RELIABILITY IMPROVEMENTS**

J. Burchfiel - BBN-TENEX  
B. Cosell - BBN-NET  
R. Tomlinson - BBN-TENEX  
D. Walden - BBN-NET

During the past months we have felt strong pressure to improve the reliability of TIP/TENEX network connection as improvement in the reliability of users' connections between TENEXes and TIPs would have major impact on the appearance of overall network reliability due to the large number and high visibility of TENEXes and TIPs. Despite the emphasis on TIP/TENEX interaction, all work done applies equally well to interactions between Hosts of any type.

The remainder of this RFC gives a sketch of our plan for improving the reliability of connections between TIPs and TENEXes. Major portions of this plan have already been implemented (TIP version 322; TENEX version 1.32) and are now undergoing final test prior to release throughout the network. Completion of the implementation of the plan is expected in the next quarter.

Our plan for improving the reliability of TIP/TENEX connections is concerned with obtaining and maintaining TIP/TENEX connections, gracefully recovering from lost connections, and providing clear messages to the user whenever the state of his connection changes.

When a TIP user attempts to open a connection to any Host, the Host may be down. In this case it would be helpful to provide the user with information about the extent of the Host's unavailability. To facilitate this, we modified the IMP program to accept and utilize information from a Host about when the Host will be back up and for what reason it is down. TENEX is to be modified to supply such information before it goes down, or through manual means, after it has gone down. When the TIP user then attempts to connect to the down TENEX, the IMP local to the TENEX returns the information about why and for how long TENEX will be down. The TIP is to be modified to report this sort of information to the user; e.g., "Host unavailable because of hardware maintenance -- expected available Tuesday at 16:30 GMT".

The TIP's logger is presently not reentrant. Thus, no single TIP user can be allowed to tie up the logger for too long at a time; and the TIP therefore enforces a timeout of arbitrary length (about 60 seconds) on logger use. However, a heavily loaded Host cannot be guaranteed always to respond within 60 seconds to a TIP login request, and at present TIP users sometimes cannot get connected to a heavily loaded TENEX. To correct this problem, the TIP logger will be made reentrant and the timeout on logger use will be eliminated.

One notorious soft spot in the Host/Host protocol which degrades the reliability of connections is the Host/Host protocol incremental allocate mechanism. Low frequency software bugs, intermittent hardware bugs, etc., can lead to the incremental allocates associated with a connection getting out of synchronization. When this happens it usually appears to the user as if the connection just "hung up". A slight addition to the Host/Host protocol to allow connection allocates to be resynchronized has been designed and implemented for both the TIP and TENEX.

---

**Preceding page blank**

**TIP/TENEX RELIABILITY IMPROVEMENTS**  
**RFC 636, NIC 30490 (June 10, 1974)**

TENEX has a number of internal consistency checks (called "bughalts") which occasionally cause TENEX to halt. Frequently, after diagnosis by system personnel, TENEX can be made to proceed without loss from the viewpoint of local users. A mechanism is being provided which allows TENEX to proceed in this case from the point of view of TIP users of TENEX.

The appropriate mechanism entails the following: TENEX will not drop its ready line during a bughalt (from which TENEX can usually proceed successfully), nor will it clear its NCP tables and abort all connections. Instead, after a bughalt TENEX will: discard the message it is currently receiving, as the IMP has returned an Incomplete Transmission to the source for this message; reinitialize the interface to the IMP; and resynchronize, on all connections possible, Host/Host protocol allocate inconsistencies due to lost messages, RFNMs etc. The latter is done with the same mechanism described above. This procedure is not guaranteed to save all data -- a tiny bit may be lost -- but this is of secondary importance to maintaining the connection over the TENEX bughalt.

The TIP user must be kept fully informed as TENEX halts and then continues. Therefore, the TIP has been modified to report "Host not responding -- connection suspended" when it senses that TENEX has halted (it does this by properly interpreting messages returned by the destination IMP). When TENEX resumes service after proceeding from a bughalt, the above procedure notifies the TIP that service is restored, and the TIP has been modified to report "Service resumed" to all users of that Host.

On the other hand, the service interruption may not be proceedable and TENEX may have to do a total system reload and restart. In this case TENEX will clear its NCP connection tables and send a Host/Host protocol reset command to all other Hosts. On receiving this reset command, the TIP will report "Host reset -- connection closed" to all users of that Host with suspended connections. The TIP user can then re-login to the TENEX or to some other Host.

Of course, the user may not have the patience to wait for service to resume after a TENEX bughalt. Instead, he may unilaterally choose to connect to some other Host, ignoring the previously suspended connection. If TENEX is then able to proceed, its NCP will still think its connection to the TIP is good and suitable for use. Thus, we have a connection which the TIP thinks is closed and TENEX thinks is open, a phenomenon known as the "half-closed connection". An automatic procedure for cleanly completing the closing of such a connection has been specified and implemented for the TIP and TENEX.

Since TENEX will maintain connections across service interruptions, the TIP user will be required to take the security procedure telling the TIP to "forget" his suspended connection before abandoning his terminal. The command @H 0 (for example) will guarantee that his connection will not be reestablished on resumption of service. Otherwise, his job would be left at the mercy of anyone who acquires that terminal.

An appendix follows which describes the Host/Host protocol changes made. These changes are backward compatible (with the exception that Hosts which have not implemented these changes will sometimes receive unrecognizable Host/Host protocol commands which they presumably discard without suffering harm). These protocol changes are ad hoc in nature but in light of their backward compatibility and potential utility, ARPA okayed their addition to the TIP and TENEX NCPs without (we believe) any implication that other Hosts have to implement them (although we would encourage their widespread implementation).

TIP/TENEX RELIABILITY IMPROVEMENTS  
RFC 636, NIC 30490 (June 10, 1974)

## APPENDIX – AD HOC CHANGE TO HOST-HOST PROTOCOL

### A.1 INTRODUCTION

The current Host–Host protocol (NIC #8246) contains no provisions for resynchronizing the status information kept at the two ends of each connection. In particular, if either host suffers a service interruption, or if a control message is lost or corrupted in an interface or in the subnet, the status information at the two ends of the connection will be inconsistent.

Since the current protocol provides no way to correct this condition, the NCPs at the two ends stay "confused" forever. An occasional frustrating symptom of this effect is the "lost allocate" phenomenon, where the receiving NCP believes that it has bit and message allocations outstanding, while the sending NCP believes that it does not have any allocation. As a result, information flow over that connection can never be restarted.

Use of the Host–Host RST (reset) command is inappropriate here, as it destroys all connections between the two hosts. What is needed is a way to resynchronize only the affected connection without disturbing any others.

A second troublesome symptom of inconsistency in status information is the "half-closed" connection: after a service interruption or network partitioning, one NCP may believe that a connection is still open, while the other believes that the connection is closed (does not exist). When such an inconsistency is discovered, the "open" end of the connection should be closed.

### A.2 THE RAR, RAS AND RAP COMMANDS

To achieve resynchronization of allocation, we add the following three commands to the host–host protocol.

8 bits    8 bits  
-----  
16 ! RAR    ! link !  
-----

Reset Allocation by Receiver

8 bits    8 bits  
-----  
17 ! RAS    ! link !  
-----

Reset Allocation by Sender

8 bits    8 bits  
-----  
20 ! RAP    ! link !  
-----

Reset Allocation Please

The RAS command is sent from the Host sending on "link" to the Host receiving on "link". This command may be sent whenever the sending Host desires to resynch the status information associated with the connection (and doesn't have a message in transit through the network). Some circumstances in which the sending Host may choose to do this are:

**TIP/TENEX RELIABILITY IMPROVEMENTS**  
**RFC 636, NIC 30490 (June 10, 1974)**

1. After a timeout when there is traffic to move but no allocation (assumes that an allocation has been lost)
2. When an inconsistent event occurs associated with that connection (e.g. an outstanding allocation in excess of  $2^{32}$  bits or  $2^{16}$  messages)
3. After the sending host has suffered an interruption of network service
4. In response to a RAP (see below).

The RAR command is sent from the Host receiving on "link" to the Host sending on "link" in response to an RAS. It marks the completion of the connection resynchronization. When the RAR is returned the connection is in the known state of having no messages in transit in either direction and the allocations are zero. The receiving Host may then start afresh with a new allocation and normal message transmission can proceed. Since the RAR may be sent ONLY in response to an RAS, there are no races in the resynchronization. All of the initiative lies with the sending Host.

If the receiving Host detects an anomalous situation, however, there is no way to inform the sending Host that a resynchronization is desirable. For this purpose, the RAP command is provided. It constitutes a "suggestion" on the part of the receiving Host that the sending Host resynchronize; the sending Host is free to honor it or not as it sees fit. Since there is no obligatory response to a RAP, the receiving Host may send them as frequently as it chooses and no harm can occur. For example, if a message in excess of the allocate arrives, the receiving Host might send RAPs every few seconds until the sending Host replies with no fears of races if one or more RAPs pass a RAS in the network.

#### A.3 RESYNCHRONIZATION PROCEDURE

The resynchronization sequence below may be initiated only by the sender either for internally generated reasons or upon the receipt of a RAP.

- a) Sender – decision to resynch
  1. Set state to "Wait-for-RAR" (Defer transmission of message).
  2. Wait until no RFNM outstanding
  3. Send RAS
  4. Zero allocation
  5. Ignore allocates until RAR received
  6. Set state to "Open" (Resume normal message transmission subject to flow control).
- b) Receiver – receipt of RAS
  1. Send RAR
  2. Zero allocation
  3. Send a new allocation

When the sender is in the "Wait-for-RAR" state it is not permitted to send new regular messages. (Note that steps 4 and 5 will insure this in the normal course of events.) With the return of the RAR the pipeline contains no messages and no allocates, the outstanding allocation variables at both ends are forced into agreement by setting them both to zero. The receiver will then reconsider bit and message allocation, and send an ALL command for any allocation it cares to do.

TIP/TENEX RELIABILITY IMPROVEMENTS  
RFC 636, NIC 30490 (June 10, 1974)

#### A.4 THE PROBLEM OF HALF-CLOSED CONNECTIONS

The above procedures provide a way to resynchronize a connection after a brief lapse by a communications component, which results in lost messages or allocates for an open connection.

A longer and more severe interruption of communication may result from a partitioning of the subnet or from a service interruption on one of the communicating hosts. It is undesirable to tie up resources indefinitely under such circumstances, so the user is provided with the option of freeing up these resources (including himself) by unilaterally dissolving the connection. Here "unilaterally" means sending the CLS command and closing the connection without receiving the CLS acknowledgment. Note that this is legal only if the subnet indicates that the destination is dead.

When service is restored after such an interruption, the status information at the two ends of the connection is out of synchronization. One end believes that the connection is open, and may proceed to use the connection. The disconnecting end believes that the connection is closed (does not exist), and may proceed to re-initialize communication by opening a new connection (RTS or STR command) using the same socket pair or same link.

The resynchronization needed here is to properly close the open end of the connection when the inconsistency is detected. We will accomplish this by specifying consistency checks and adding a new pair of commands.

#### A.5 THE NXS AND NXR COMMANDS

The "missing CLS" situation described above can manifest itself in two ways. The first way involves action taken by the NCP at the "open" end of the connection. It may continue to send regular messages on the link of the half-closed connection, or control messages referencing its link. The closed end should respond with an NXS if the message referred to a non-existent transmit link (e.g. was an ALL) or NXR if the message referred to a non-existent receive link (e.g. a data message). On receipt of such an NXS or NXR message, the NCP at the "open" end should close the connection by modifying its tables (without sending any CLS command) thereby bringing both ends into agreement.

8 bits	8 bits
-----	
21 ! NXR	! link !
-----	
Non-existent Receive Link	
8 bits	8 bits
-----	
22 ! NXS	! link !
-----	
Non-existent Send Link	

#### A.6 CONSISTENCY CHECKS

A second way this inconsistency can show up involves actions initiated by the NCP at the "closed" end. It may (thinking the connection is closed) send an STR or RTS to reopen the

**TIP/TENEX RELIABILITY IMPROVEMENTS**  
**RFC 636, NIC 30490 (June 10, 1974)**

connection. The NCP at the "open" end should detect the inconsistency when it receives such an RTS or STR command, because it specifies the same socket pair as an existing open connection, or, in the case of an RTS, the same link. In this case, the NCP at the "open" end should close the connection (without sending any CLS command) to bring the two ends into agreement before responding to the RTS/STR.

**A.7 CONCLUSION**

The scheme presented in Section A.2 to resynchronize allocation has one very important property: the data stream is preserved through the exchange. Since no data is lost, it is safe to initiate resynchronization from either end at any time. When in doubt, resynchronize.

The consistency checks for RTS and STR, and the NXR and NXS commands provide the synchronization needed to complete the closing of "half-closed" connections.

The protocol changes above will make the host-host protocol far more robust, in that useful work can continue in spite of lapses by the communications components.

**TRANSMISSION CONTROL PROTOCOL  
NIC 29628 (Dec. 13, 1977)**

J. Postel (USC-ISI)  
NIC 29628 (Dec. 13, 1977)

**TRANSMISSION CONTROL PROTOCOL (TCP)**

The Transmission Control Protocol (TCP) is intended to be used as a host-to-host protocol between hosts in a computer network or (more especially) between hosts in interconnected computer networks. TCP is, then, an internetwork host level protocol.

TCP is being used in the ARPA-sponsored work on packet switched radio communication (PRNET) and packet switched satellite communication (SATNET), and interconnections of these experimental facilities with each other and the ARPANET. TCP implementations exist for PDP-10 TENEX systems and PDP-11 systems, and TCP implementations are under development for TOPS20, Multics, the Norwegian NORD-10 system, and UCLA-CCN's IBM-360/91. Some hosts in the MIT LCSNET will intercommunicate with hosts in the ARPANET using TCP.

TCP is designed to support interprocess communication by establishing full-duplex logical connections between process input/output ports. The data exchanged between processes is a stream of octets divided into variable length letters. The TCP provides for reliable communication by using end-to-end acknowledgments and checksums. TCP modules exchange control information to establish and terminate connections, to regulate the flow of data on connections, and to signal the need for urgent handling of a connection. The control information is communicated between TCP modules in the headers of messages on the data connections. There is no separate control connection.

TCP development is based upon a paper by Cerf and Kahn published in May 1974 [1]. The first specification was produced by Cerf and his students at Stanford University in December 1974 [2]. A group at SRI International working under contract to the Defense Communications Agency produced a specification for TCP for use in the AUTODIN II network in July 1976 [3]. In March of 1977 Cerf (now at ARPA) produced the specification of TCP version 2 [4]. The current specification of TCP is version 3 as documented by Cerf and Postel in January 1978 [5].

**TRANSMISSION CONTROL PROTOCOL**  
**NIC 29628 (Dec. 13, 1977)**

**REFERENCES**

1. Cerf, Vinton G. and Kahn, Robert E. A Protocol for Packet Network Intercommunication, IEEE Trans. Commun., Vol. COM-22, No. 5, 637-48 (May 1974). (An early version of this paper appeared as INWG General Note 39, IFIP Working Group 6.1, (Sept. 1973).
2. Cerf, Vinton G., Dalal, Yogen K. and Sunshine, Carl. Specification of Internet Transmission Control Program, INWG General Note 72, IFIP Working Group 6.1 (Dec. 1974), also RFC 675, NIC 31505.
3. Postel, Jonathan B., Garlick, Larry L. and Rom, Raphael. Transmission Control Protocol Specification, NLS 35938 and 35939, Augmentation Research Center, SRI International, Menlo Park, Calif., also Rept. 35938, Defense Communications Engineering Center, Reston, Va. (July 15, 1976).
4. Cerf, Vinton G. Specification of Internet Transmission Control Program - TCP (Version 2), Advanced Research Projects Agency, Arlington, Va. (March 1977).
5. Cerf, Vinton G. and Postel, Jonathan B. Specification of Internet Transmission Control Program - TCP Version 3, Information Sciences Institute, University of Southern California, Marina Del Rey, Calif. (Jan. 1978).

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**SPECIFICATIONS FOR THE  
NETWORK VOICE PROTOCOL (NVP)**

and

Appendix 1: The Definition of Tables—Set—#1 (for LPC)

Appendix 2: Implementation Recommendations

**NSC NOTE 68**  
(Revision of NSC Notes 26, 40, and 43)

Danny Cohen, USC-ISI  
January 29, 1976



NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

CONTENTS

PREFACE .....	iii
ACKNOWLEDGMENTS .....	iv
INTRODUCTION .....	1
THE CONTROL PROTOCOL .....	2
Summary of the CONTROL Messages .....	3
Definition of the CONTROL Messages .....	4
Definition of the <WHAT> and <HOW>	
Negotiation Tables .....	7
On RENEgotiation .....	9
The Header of Data Messages .....	9
THE LPC DATA PROTOCOL .....	11
EXAMPLES FOR THE CONTROL PROTOCOL .....	13
APPENDIX 1: THE DEFINITION OF TABLES-SET-#1 .....	17
General Comments .....	20
Comments on the PITCH Table .....	20
Comments on the GAIN Table .....	20
Comments on the INDEX7 Table .....	21
Comments on the INDEX6 Table .....	21
Comments on the INDEX5 Table .....	21
The PITCH Table .....	22
The GAIN Table .....	23
The INDEX7 Table .....	24
The INDEX6 Table .....	25
The INDEX5 Table .....	26
APPENDIX 2: IMPLEMENTATION RECOMMENDATIONS .....	27
REFERENCES .....	29

Preceding page blank

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

PREFACE

The major objective of ARPA's Network Secure Communications (NSC) project is to develop and demonstrate the feasibility of secure, high-quality, low-bandwidth, real-time, full-duplex (two-way) digital voice communications over packet-switched computer communications networks. This kind of communication is a very high priority military goal for all levels of command and control activities. ARPA's NSC project will supply digitized speech which can be secured by existing encryption devices. The major goal of this research is to demonstrate a digital high-quality, low-bandwidth, secure voice handling capability as part of the general military requirement for worldwide secure voice communication. The development at ISI of the Network Voice Protocol described herein is an important part of the total effort.

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**ACKNOWLEDGMENTS**

The Network Voice Protocol (NVP), was implemented first in December 1973, and has been in use since then for local and transnet real-time voice communication over the ARPANET at the following sites:

- o Information Sciences Institute, for LPC and CVSD, with a PDP-11/45 and an SPS-41.
- o Lincoln Laboratory, for LPC and CVSD, with a TX2 and the Lincoln FDP, and with a PDP-11/45 and the LDVT.
- o Culler-Harrison, Inc., for LPC, with the Culler-Harrison MP32A and AP-90.
- o Stanford Research Institute, for LPC, with a PDP-11/40 and an SPS-41.

The NVP's success in bridging the differences between the above systems is due mainly to the cooperation of many people in the ARPA-NSC community, including Jim Forgie (Lincoln Laboratory), Mike McCammon (Culler-Harrison), Steve Casner (ISI) and Paul Raveling (ISI), who participated heavily in the definition of the control protocol; John Markel (Speech Communications Research Laboratory), John Makhoul (Bolt Beranek & Newman, Inc.) and Randy Cole (USC-ISI), who participated in the definition of the data protocol. Many other people have contributed to the NVP-based effort, in both software and hardware support.



## SPECIFICATIONS FOR THE NETWORK VOICE PROTOCOL

### INTRODUCTION

Currently, computer communication networks are designed for data transfer. Since there is a growing need for communication of real-time interactive voice over computer networks, new communication discipline must be developed. The current HOST-to-HOST protocol of the ARPANET, which was designed (and optimized) for data transfer, was found unsuitable for real-time network voice communication. Therefore this Network Voice Protocol (NVP) was designed and implemented.

Important design objectives of the NVP are:

- Recovery of loss of any message without catastrophic effects. Therefore all answers have to be unambiguous, in the sense that it must be clear to which inquiry a reply refers.
- Design such that no system can tie up the resources of another system unnecessarily.
- Avoidance of end-to-end retransmission.
- Separation of control signals from data traffic.
- Separation of vocoding-dependent parts from vocoding-independent parts.
- Adaptation to the dynamic network performance.
- Optimal performance, i.e. guaranteed required bandwidth, and minimized maximum delay.
- Independence from lower level protocols.

The protocol consists of two parts:

- 1 The control protocol,
2. The data protocol.

Control messages are sent as controlled (TYPE 0/0) messages, and data messages may be sent as either controlled (TYPE 0/0) or uncontrolled (TYPE 0/3) messages (see BBN Report 1822 for definition of MESSAGE-TYPE).

Throughout this document a "word" means a "16-bit quantity".

Preceding page blank

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**THE CONTROL PROTOCOL**

Throughout this document the 12-bit MESSAGE-ID (see BBN Report 1822) is referred to as LINK (its 8 MSBs) and SUB-LINK (its 4 LSBs).

The control protocol starts with an initial connection phase on link 377 and continues on other links assigned at run time.

Four links are used for each voice communication:

- Link L will be used for control, from CALLER to ANSWERER.
- Link K will be used for control, from ANSWERER to CALLER.
- Link L+1 will be used for data, from CALLER to ANSWERER.
- Link K+1 will be used for data, from ANSWERER to CALLER.

Both L and K should be between 340 and 375 (octal). L and K need not differ.

The first message (CALLER to ANSWERER) on link 377 indicates which user wants to talk to whom and specifies K. As a response (on K), the ANSWERER either refuses the call or accepts it and assigns L.

The CALLER then calls again (this time on link L). The ANSWERER initiates a negotiation session to verify the compatibility of the two parties.

The negotiation consists of suggestions put forth by one of the parties, which are either accepted or rejected by the other party. The suggesting party in the negotiation is called the NEGOTIATION MASTER. The other party is called the NEGOTIATION SLAVE. Usually the ANSWERER is the negotiation master, unless agreed otherwise by the method described later.

If the negotiation fails, either party may terminate the call by sending a "GOODBYE". If the negotiation is successfully ended, the ANSWERER rings bells to draw human attention and sends "RINGING" to the CALLER. When the call is answered (by a human), a "READY" is sent to the CALLER and the data starts flowing (on L+1 and K+1). However, a "READY" can be sent without a preceding "RINGING".

This bell ringing occurs only after the initial call (not after renegotiation).

The assignment of L and K cannot be changed after the initial connection phase.

Only one control message can be sent in a network-message. Extra bits needed to fill the network-message are ignored.

The length of control messages should never exceed a single-packet (i.e., 1,007 data bits).

Control messages not recognized by their receiver should be ignored and should not cause any error condition resulting in termination of the connection. These messages may result from differences in implementation level between systems.

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

SUMMARY OF THE CONTROL MESSAGES

- #1 "1,<WHO>,<WHOM>,K"
- #2 "2,<CODE>" or only "2"
- #3 "3,<WHAT>,<N>,<HOW(1),...HOW(N)>"
- #4 "4,<WHAT>,<HOW>"
- #5 "5,<WHAT>,<HOW>" or only "5,<WHAT>"
- #6 "6,L" or only "6"
- #7 "7"
- #8 "8"
- #9 "9"
- #10 "10,<ID>"
- #11 "11,<ID>"
- #12 "12,<IM>"
- #13 "13,<YM>,<OK>"

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**DEFINITION OF THE CONTROL MESSAGES**

**#1 CALLING (on 377 and L)**

This call is issued first on link 377 and later on link L. Its format is "1,<WHO>,<WHOM>,K", where <WHO> and <WHOM> are words which identify respectively the calling party and the party that is being called, and K is as defined above. The format of the <WHO> and <WHOM> is:

(HHIIIII|XXXXXXXX)

where HH are 2 bits identifying the HOST, followed by 6 bits identifying the IMP, followed by 8 bits identifying the extension (needed because there may be more than one communication unit on the same HOST).

The system which sends this message is defined as the CALLER, and the other system is defined as the ANSWERER.

**#2 GOODBYE (TERMINATION, on L or K)**

This message has the purpose of terminating calls at any stage.

ICP can be terminated (on K) either negatively by sending either a single word "2" ("GOODBYE") or the two words "2,<CODE>", or positively by sending the two words "6,L", as described later.

After the initial connection phase, calls can be terminated by either the CALLER (on L) or the ANSWERER (on K). This termination has two words: "2,<CODE>", where <CODE> is the reason for the termination, as specified here:

0. Other than the following.
  1. I am busy.
  2. I am not authorized to talk with you.
  3. Request of my user.
  4. We believe you are down.
  5. Systems incompatibility (NEGOTIATION failure).
  6. We have problems.
  7. I am in a conference now.
  8. You made a protocol error.

**#3 NEGOTIATION INQUIRY (on L or K)**

Sent by the NEGOTIATION MASTER for compatibility verification. The format is:

"3,<WHAT>,<LIST-LENGTH>,<HOW-LIST>", meaning

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

"CAN-YOU-DO,<WHAT>,<LIST-LENGTH>,<HOW-LIST>".

The <HOW-LIST> is a list of pointers into agreed-upon tables, as shown below.

#4 POSITIVE NEGOTIATION RESPONSE (on L or K)

Sent by the NEGOTIATION SLAVE in response to a NEGOTIATION INQUIRY. The format is:

"4,<WHAT>,<HOW>", meaning: "I-CAN-DO,<WHAT>,<HOW>".

#5 NEGATIVE NEGOTIATION RESPONSE (on L or K)

Sent by the NEGOTIATION SLAVE in response to a NEGOTIATION INQUIRY. The format is either:

"5,<WHAT>,0", meaning "I-CAN'T-DO-<WHAT>-IN-ANY-OF-THESE-WAYS",

or: "5,<WHAT>,N", meaning inability to accept any of the options offered in the INQUIRY, but using "N" as a suggestion to the ANSWERER about another possibility. Examples are presented later in this report.

#6 READY (on L or K)

Sent by either party to indicate readiness to accept data. Its format is "6,L" in the reply to the initial call, and "6" thereafter.

#7 NOT READY (on L or K)

Sent by either party to indicate unreadiness to accept data. It is always a single word: "7".

#8 INQUIRY (on L or K)

Sent by either party to inquire about the status of the other. It is always a single word: "8". It is answered by #6, #7, or #9.

#9 RINGING (on K)

Sent by the ANSWERER after the negotiations have been successfully terminated and human permission is needed to proceed further. The ringing will continue for 10 seconds, and then stop, UNLESS a #8 is received. This message is always a single word: "9".

#10 ECHO REQUEST (on L or K)

Sent by whichever party is interested in measuring the network delays. Its only purpose is to be echoed immediately. The format is "10,<ID>", where <ID> is any word used to identify the ECHO.

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**#11 ECHO (on L or K)**

Sent in response to ECHO REQUEST. The format is "11,<ID>", where <ID> is the word specified by #10. The implementation of this feature is not compulsory, and no connection should be terminated due to lack of response to ECHO-REQUEST.

**#12 RENEGOTIATION REQUEST (on L or K)**

Can be sent by either party at ANY stage after LINKS are agreed upon. This message consists of the two words "12,<IM>". If the word <IM> (for I MASTER) is non-zero, the sender of this message requests to be the NEGOTIATION MASTER. If it is zero, the receiver of this message is requested to be the NEGOTIATION MASTER. Renegotiation is described later.

**#13 RENEGOTIATION APPROVAL (on L or K)**

This message may be sent by either party in response to RENEGOTIATION REQUEST. It consists of the three words "13,<YM>,<OK>". If <OK> is non-zero, this is a positive acknowledgment (approval). If it is zero, this is a negative acknowledgment (i.e., refusal). <YM> is set to be equal to the <IM> of #12, for identification purposes.

Messages #7, #8, and #9 are always a single word. Messages #1, #3, #4, and #5 are several words long. Messages #2 and #6 are either a single word or two words long. #10, #11 and #12 are always 2 words long. Message #13 is always 3 words long. Message #1 is always 4 words long.

Message #1 is sent only by the CALLER, #3 only by the NEGOTIATION MASTER, and #4 and #5 only by the NEGOTIATION SLAVE. Message #9 is sent only by the ANSWERER. All the other control messages may be sent by either party.

The last <HOW> which was both suggested by the NEGOTIATION MASTER (in #3) and accepted by the NEGOTIATION SLAVE (in #4) for each <WHAT> is assumed to be in use.

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

DEFINITION OF THE <WHAT> AND <HOW> NEGOTIATION TABLES:

<WHAT>	<HOW>
1. VOCODING	* 1. LPC + 2. CVSD 3. RELP 4. DELCO
2. SAMPLE PERIOD  (in microseconds)	N. N (*150) (+62)
3. VERSION	* 1. V1 (see definition below) + 2. V2 (see definition below)
4. MAX MSG LENGTH (in bits)	
NVP header included (32 bits) but not HOST/IMP leader and not HOST/IMP padding	N. N (*976 and +976)
5. If LPC:	
Degree	N. For N coefficients (*10)
If CVSD:	
Time Constant (in milliseconds)	N. N (+50)
6. Samples per Parcel	N. N (*128) (+224)
7. If LPC:	
Acoustic Coding	* 1. SIMPLE (see below) 2. OPTIMIZED
8. If LPC:	
Info Coding	* 1. SIMPLE (see below) 2. OPTIMIZED
9. If LPC:	
Pre-emphasis 1 - mu x [Z**-1] N = 64 x mu	N. N (*58, for mu = 58/64 = 0.90625)

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

10. If LPC:

Table-set

N. N (\*1)

See definition of Set #1  
in Appendix 1

(\* indicates recommended options for LPC)  
(+ indicates recommended options for CVSD)

No parameter (<WHAT>) should be inquired about by the NEGOTIATION MASTER if some option (<HOW>) for it has been previously accepted by the NEGOTIATION SLAVE implicitly in the "VERSION". The purpose of this restriction is to avoid a possible conflict between individual parameters and the VERSION-option.

Version 1 (V1) is defined as:

1-1      LPC  
2-150    150 microseconds sampling  
3-1      V1  
5-10     10 coefficients  
6-128    128 samples per parcel  
7-1      SIMPLE acoustic coding  
8-1      SIMPLE information coding  
9-58     mu = 58/64 = 0.90625  
10-1     Tables set #1

Version 2 (V2) is defined as:

1-2      CVSD  
2-62     62 microseconds sampling (16 KHz sampling)  
3-2      V2  
5-50     50 msec time constant  
6-192    192 samples per parcel

Note that this defines every negotiated parameter, except MAX MSG LENGTH.

SIMPLE and OPTIMIZED codings will be described below in Section 3.

All the negotiation is managed by the NEGOTIATION MASTER, who decides how much negotiation is needed, and what to do in case some discrepancy (incompatibility) is discovered: either to try alternative options or to abort the connection. Upon completion of successful negotiation, the NEGOTIATION MASTER sends either #9 (RINGING) only if it is the ANSWERER and if this is an initial connection, else it sends #6 (READY-FOR-DATA), and probably inquires with #8 about the readiness of the other party. The inquiries (#8) before the successful completion of the negotiation are ignored. However, these inquiries after the first RINGING (#9) and before the first READY (#6) are needed to keep the ANSWERER ringing.

Note that the negotiation process can be shortened by using the VERSION option, as shown in the examples that follow.

## ON RENEGOTIATION

At any stage after links are agreed upon, either party might request a RENEGOTIATION. If the request is approved by the other party, either party might become the NEGOTIATION MASTER, depending on the type of renegotiation request. When renegotiation starts, no previously negotiated agreements (except LINK numbers) hold, and all items have to be renegotiated from scratch. Note that renegotiation may entirely replace the negotiation phase and allows the CALLER to be the NEGOTIATION MASTER.

Upon issuance (or reception) of RENEGOTIATION REQUEST, all data messages are ignored until the positive indication of the successful completion of the renegotiation (#6).

After the completion of renegotiation, the frame-count (see the section on MESSAGE-HEADER) may be reset to zero.

## THE HEADER OF DATA MESSAGES

Data messages are the messages which contain vocoded speech. The first 32 bits of each data message is the MESSAGE-HEADER, which carries sequence and timing information as described below.

For each vocoding scheme a "FRAME" is defined as the transmission interval (as agreed upon at the negotiation stage in <WHAT#6>). Since this interval is defined by the number of samples, its duration can be found by multiplying the sampling period <WHAT#2> by the interval length (in samples) <WHAT#6>. For example, in V1 the sampling period is 150 microseconds and the transmission interval is 128 samples, which yields:

$$128 \times 150 \text{ microseconds} = 19.2 \text{ milliseconds.}$$

The data describing a FRAME is called a PARCEL. Each parcel has a serial number. The first parcel created after the completion of the negotiation (or every RENEGOTIATION) has the serial number zero. Each message contains an integral number of parcels.

The serial number of the first parcel in the message is put in the first 16 bits of the message and is referred to as the MESSAGE-TIME-STAMP. Note that this time stamp is synchronized with the data stream. Note also that these 16 bits are actually the third word of the message, following the 2 words used as IMP-to-HOST leader (see BBN Report 1822).

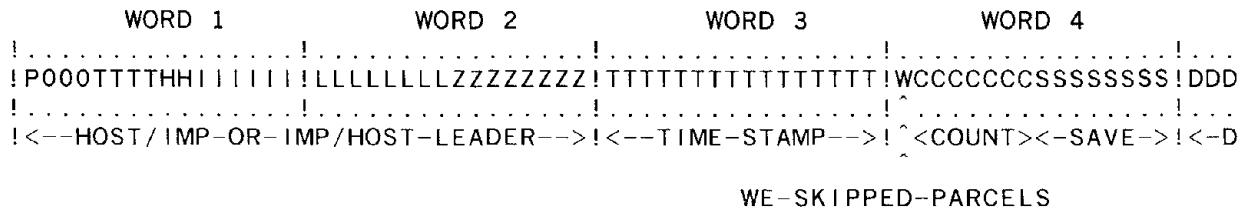
The next bit in the header is the WE-SKIPPED-PARCELS bit, which is described later. The next 7 bits tell how many parcels there are in the message; this number is called the COUNT, or the PARCEL-COUNT.

Note that if message number N has the time stamp T(N) and the count C(N), then T(N+1) must be greater than or equal to T(N)+C(N). Usually T(N+1) = T(N)+C(N), unless the XMTR decided not to send some parcels due to silence. If this happens then the WE-SKIPPED-PARCELS bit is set to ONE, else it is set to ZERO. Hence, if T(N+1) is found by the RCVR to be greater than T(N)+C(N) and the WE-SKIPPED-PARCELS is zero, some message must be lost.

Note that by definition the time stamps on messages monotonically increase, except for wrap-around.

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

The message header structure is illustrated by the following diagram:



P = PRIORITY (one bit = 1)

T = MESSAGE TYPE (4 bits = 0011)

L = link ("L" OR "K", 8 bits, greater than 337 octal)

D = data bits (from here to the end of the message)

ZZZZZZZZ = 8 ZERO bits

HH!!!!!! = HOST (8 bits, destination or source)

CCCCCC = parcel COUNT (7 bits)

SSSSSSSS = 8 bits saved for future applications

TTTTTTTTTTTTTT = TIME STAMP (16 bits)

The first parcel sent by either party after the NEGOTIATION or RENEGOTIATION should have the serial number set to zero.

During silence periods, the XMTR might send a "6" or "7" message periodically. If it does not do so, the RCVR might interrogate the livelihood of the XMTR by sending periodically "8" ("ARE-YOU-THERE?") or #10 (ECHO-REQUEST) messages.

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

### THE LPC DATA PROTOCOL

The DATA sent at each transmission interval is called a PARCEL.

Network messages always contain an integral number of PARCELS.

There are two independent issues in the coding. One is, obviously, the acoustic coding, i.e., which parameters have to be transmitted. SIMPLE acoustic coding is sending all the parameters at every transmission interval. OPTIMIZED acoustic coding sends only as little as is acoustically needed. DELCO is an example of OPTIMIZED acoustic coding.

In this document only the format of the SIMPLE acoustic coding is defined.

All the transmitted parameters are sent as pointers into agreed-upon tables. These tables are defined as two lists of values. The transmitter table  $[X(J)]$  is used in the following way: The value V is coded as the code J if  $X(J-1) < V \leq X(J)$ . The receiver table  $[R(J)]$  is used to retrieve the value R(J) if the code J was received.  $X(-1)$  is implicitly defined as minus-infinity, and  $X(J_{max})$  is explicitly defined as plus-infinity.

For each parameter,  $[X(J)]$  and  $[R(J)]$  may be defined independently.

The second coding issue is the information coding technique. The SIMPLE (information-wise) way of sending the information is to use binary coding for the codes representing the parameters. The OPTIMIZED way is to compute distributions for each parameter and to define the appropriate coding. It is very probable that the PITCH and GAIN will be decoded absolutely in the first PARCEL of each message, and incrementally thereafter.

At present, only the SIMPLE (information-wise) coding is used.

The details of the LPC data protocol and its Tables-Set-#1 can be found in Appendix 1.

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

Following is the definition for the format of the SIMPLE-SIMPLE coding, according to Tables-Set-#1:

For each parcel:

PITCH	6 bits (PITCH=0 for UNVOICED)
GAIN	5 bits
I(1)	7 bits
I(2)	7 bits
I(3)	6 bits
I(4)	6 bits
I(5)	5 bits
I(6)	5 bits
I(7)	5 bits
I(8)	5 bits
I(9)	5 bits
I(10)	5 bits

where each of the I(j) is an index for inverse sine coding. If  $K(j)=\arcsin(\Theta(j))$  and N bits are assigned for its transmission, then  $I(j)=(\Theta(j)/\pi)*2^{**N}$ .

Hence at each transmission interval (128 samples times 150 microseconds) 67 bits are sent, which results in a data rate of 3490 bps. Since this bandwidth is well within the capabilities of the network, SIMPLE-SIMPLE coding is used, which requires the least computation by the hosts. Note that this data rate is a peak rate, without the use of silence.

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

EXAMPLES FOR THE CONTROL PROTOCOL

Here is an example for a connection:

(377) C: 1,<WHO>,<WHOM>,340      Please talk to me on 340/341.  
(340) A: 2,1                                  I refuse, since I'm busy.

Another example:

(377) C: 1,<WHO>,<WHOM>,360      Please talk to me on 360/361.  
(360) A: 6,350                                  OK. You talk to me on 350/351.  
(350) C: 1,<WHO>,<WHOM>                    I want to talk to you.  
(360) A: 3,1,1,2                                Can you do CVSD? (ANSWERER tries to be the NEGOTIATION MASTER)  
(350) C: 12,1                                    I want to be it.  
(360) A: 13,1                                    That's OK with me.  
(350) C: 3,1,1,2                                Can you do CVSD?  
(360) A: 5,1,1                                    No, but I can do LPC.  
(350) C: 3,1,1,3                                Can you do RELP?  
(360) A: 5,1,1                                    No, but I can do LPC.  
(350) C: 3,1,1,1                                How about LPC?  
(360) A: 4,1,1                                    LPC is fine with me.  
(350) C: 3,2,1,150                            Can you use 150 microseconds sampling?  
(360) A: 4,2,150                                I can use 150 microseconds.  
(350) C: 3,4,3,976,1040,2016            Can you use 976, 1040, or 2016 bits/msg?  
(360) A: 4,4,976                                I can use 976.  
(350) C: 3,5,1,10                              Can you send 10 coefficients?  
(360) A: 4,5,10                                 I can send 10.  
(350) C: 3,6,1,64                              Can you use a 64 sample transmission?  
(360) A: 4,6,64                                 I can use 64.  
(350) C: 3,7,2,1,2                            SIMPLE or OPTIMIZED acoustic coding?  
(360) A: 4,7,2                                    OPTIMIZED!

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

(350) C: 3,8,1,1 Can you do SIMPLE info coding?  
(360) A: 4,8,1 I can do SIMPLE.  
(350) C: 3,9,1,58 mu = 0.90625?  
(360) A: 4,9,58 Fine with me.  
(350) C: 3,10,1 Table set #1?  
(360) A: 4,10,1 Of course!  
(350) C: 6 I am ready. (Note: No "RINGING" sent)  
(350) C: 8 And you?  
(360) A: 6 I am ready, too.  
..... Data is exchanged now,  
..... on 351 and 361.  
(350) C: 10,1234 Echo it, please.  
(360) A: 11,1234 Here it comes!  
.....  
(360) A: 10,3333 Now ANSWERER wants to measure  
(350) C: 11,3333 ...the delays, too.  
.....  
(???) X: 2,3 Termination by either user.

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

Another example:

(377) C: 1,<WHO>,<WHOM>,360	Please talk to me on 360/361.
(360) A: 6,340	Fine. You send on 340/341.
(340) C: 1,<WHO>,<WHOM>	I want to talk to you.
(360) A: 3,3,1,1	Can you use V1?
(340) C: 4,3,1	Yes, V1 is OK.
(360) A: 3,4,1,1984	Can you use up to 1984 bits/msg?
(340) C: 5,4,976	No, but I can use 976.
(360) A: 3,4,1,976	Can you use up to 976 bits/msg?
(340) C: 4,4,976	I can use 976.
(360) A: 9	Ringing (note how short this negotiation is!!).
.....	
(340) C: 8	Still there?
(360) A: 9	Still ringing.
.....	
(340) C: 8	Still there?
(360) A: 9	Still ringing.
.....	
(340) C: 8	How about it?
(360) A: 9	Still ringing.
(340) C: 2	Forget it! (No reason given.)



**NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)**

**APPENDIX 1**

**THE DEFINITION OF:**

**TABLES-SET-#1**

by

John D. Markel

Speech Communication Research Laboratory

Santa Barbara, California

**Preceding page blank**



**NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)**

**TABLES-SET-#1**

This set includes tables for:

PITCH - 64 values, PITCH table  
GAIN - 32 values, GAIN table  
I( 1) - 128 values, INDEX7 table  
I( 2) - 128 values, INDEX7 table  
I( 3) - 64 values, INDEX6 table  
I( 4) - 64 values, INDEX6 table  
I( 5) - 32 values, INDEX5 table  
I( 6) - 32 values, INDEX5 table  
I( 7) - 32 values, INDEX5 table  
I( 8) - 32 values, INDEX5 table  
I( 9) - 32 values, INDEX5 table  
I(10) - 32 values, INDEX5 table

These tables are defined specifically for a sampling period of 150 microseconds.

---

**Preceding page blank**

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**GENERAL COMMENTS**

The following tables are arranged in three columns,  $[X(j)]$ ,  $[j]$ , and  $[R(j)]$ . Note that the entries in the  $[X(j)]$  column are half a step off the other columns. This is to indicate that INTERVALS from X-domain (pitch, gain, and the Ks) are mapped into CODES  $[j]$ , which are transmitted over the network, to be translated by the receiver into the  $[R(j)]$ . These intervals are defined as OPEN-CLOSE intervals. For example, the PITCH value (at the transmitter) of 4131 belongs to the interval "(4024,4131]", hence it is coded as  $j=6$  which is mapped by the receiver to the value 21. Similarly, the value of 2400 for INDEX7 is found to belong to the interval "(2009,2811]", coded into the CODE 3 and mapped back into 2411.

Note that if N bits are used by a certain CODE, then there are  $2^{**N+1}$  entries in the X-table, but only  $2^{**N}$  entries in the R-table.

The transformation values used for PITCH, GAIN, and the K-parameters (in the X- and R-tables) are as defined in NSC Note 42.

Values above and below the range of the X-table are mapped into the maximum and minimum table indices, respectively.

Note that  $R(J)$  of INDEX5 is identical to  $R(2J)$  of INDEX6, and that  $R(J)$  of INDEX6 is identical to  $R(2J)$  of INDEX7. Therefore, it is possible to store only the R-table of INDEX7, without the R-tables of INDEX5 and INDEX6.

In the SPS-41 implementation there is no need to store any R-table for the K-parameters. The transmitted index can be used directly (with the appropriate scaling) as an index into the SPS built-in TRIG tables.

**COMMENTS ON THE PITCH TABLE**

The level  $J=0$  defines the UNVOICED condition. The receiver maps it into the number of samples per frame (here 128).

This PITCH table differs significantly from previous tables and supersedes the table published in NSC Note 36. Details of the calculation of the table can be found in NSC Note 42. Immediate questions should be referred to John Markel.

**COMMENTS ON THE GAIN TABLE**

The level  $J=0$  defines absolute silence.

This table is designed for a maximum of 12-bit A/D input, and allows for a dynamic range of 43.5 dB.

NSC Notes 36, 45, 56 and 58 supply background for the GAIN table. Gain is the energy of the pre-emphasized, windowed signal.

**NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)**

This table is the NEW GAIN table. NSC Notes 56 and 58 explain the reasoning behind the NEW GAIN.

**COMMENTS ON THE INDEX7 TABLE**

Positive values are coded into the range [0–63, decimal]. Negative values are coded into the 7-bits two's complement of the codes of their absolute value [65–127, decimal].

Note that all values  $-403 < V < 403$  are coded as (and mapped into) 0. Note also that the code -64 (100 octal) is never used.

In SPS-41 implementation, the R-table is not needed, since TRIG(2J) is the needed value R(J).

**COMMENTS ON THE INDEX6 TABLE**

Positive values are coded into the range [0–31, decimal]. Negative values are coded into the 6-bits two's complement of the codes of their absolute values [33–63, decimal].

Note that all values  $-805 < V < 805$  are coded as (and mapped into) 0. Note also that the code -32 (40 octal) is never used.

In SPS-41 implementation, the R-table is not needed, since TRIG(4J) is the needed value R(J).

**COMMENTS ON THE INDEX5 TABLE**

Positive numbers are coded into the range [0–15, decimal]. Negative numbers are coded into the 5-bits two's complement of their absolute values, i.e., [17–31, decimal].

Note that all values  $-1609 < V < 1609$  are coded as (and mapped into) 0. Note also that the code -16 (20 octal) is never used.

In SPS-41 implementation, the R-table is not needed, since TRIG(8J) is the needed value R(J).

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

THE PITCH TABLE (as of 10-29-74)

X(J)	J	R(J)	X(J)	J	R(J)	X(J)	J	R(J)
0	0	128*	6002	21	33	10770	42	61
0	1	18	6168	22	34	11080	43	63
3630	2	19	6338	23	35	11399	44	65
3724	3	19	6515	24	36	11728	45	67
3821	4	20	6696	25	37	12067	46	69
3921	5	20	6883	26	38	12417	47	71
4024	6	21	7075	27	39	12776	48	73
4131	7	22	7274	28	40	13147	49	75
4240	8	22	7478	29	41	13529	50	77
4353	9	23	7689	30	43	13922	51	80
4469	10	24	7905	31	44	14327	52	82
4588	11	24	8129	32	45	14745	53	85
4711	12	25	8359	33	47	15175	54	87
4838	13	26	8596	34	48	15618	55	90
4969	14	27	8840	35	50	16075	56	93
5104	15	27	9092	36	51	16545	57	95
5242	16	28	9351	37	53	17029	58	98
5385	17	29	9618	38	54	17529	59	101
5533	18	30	9894	39	56	18043	60	104
5684	19	31	10177	40	57	18572	61	107
5841	20	32	10469	41	59	19118	62	111
6002			10770			19681		
						infinity	63	114

Note: This table has only 58 different intervals defined, since 5 values are repeated in the R(j) table.

\* This value is the "Transmission Interval" (measured in samples) as defined in item #6 of the NEGOTIATION.

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**THE GAIN TABLE (as of 9-17-75)**

X(J)	J	R(J)	X(J)	J	R(J)
0	0	0	225	16	245
20	1	20	266	17	289
22	2	24	315	18	342
26	3	28	372	19	404
30	4	33	439	20	478
36	5	39	519	21	565
42	6	46	614	22	667
50	7	54	725	23	789
59	8	64	857	24	932
70	9	76	1013	25	1101
83	10	90	1197	26	1301
98	11	106	1415	27	1538
116	12	126	1672	28	1818
137	13	148	1976	29	2148
161	14	175	2335	30	2539
191	15	207	2760	31	3000
255			infinity		

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**INDEX7 TABLE (as of 9-23-74)**

X(J)	J	R(J)	X(J)	J	R(J)	X(J)	J	R(J)
0	0	0	15800	21	16151	27897	42	28106
402	1	804	16500	22	16846	28311	43	28511
1206	2	1608	17190	23	17531	28707	44	28899
2009	3	2411	17869	24	18205	29086	45	29269
2811	4	3212	18538	25	18868	29448	46	29622
3612	5	4011	19195	26	19520	29792	47	29957
4410	6	4808	19841	27	20160	30118	48	30274
5205	7	5602	20475	28	20788	30425	49	30572
5998	8	6393	21097	29	21403	30715	50	30853
6787	9	7180	21706	30	22006	30986	51	31114
7571	10	7962	22302	31	22595	31238	52	31357
8351	11	8740	22884	32	23170	31471	53	31581
9127	12	9512	23453	33	23732	31686	54	31786
9896	13	10279	24008	34	24279	31881	55	31972
10660	14	11039	24548	35	24812	32058	56	32138
11417	15	11793	25073	36	25330	32214	57	32286
12167	16	12540	25583	37	25833	32352	58	32413
12910	17	13279	26078	38	26320	32470	59	32522
13646	18	14010	26557	39	26791	32568	60	32610
14373	19	14733	27020	40	27246	32647	61	32679
15091	20	15447	27467	41	27684	32706	62	32729
15800			27897			32746	63	32758
						infinity		

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

INDEX6 TABLE (as of 9-23-74)

X(J)	J	R(J)	X(J)	J	R(J)
0	0	0	22595	16	23170
804	1	1608	23732	17	24279
2411	2	3212	24812	18	25330
4011	3	4808	25833	19	26320
5602	4	6393	26791	20	27246
7180	5	7962	27684	21	28106
8740	6	9512	28511	22	28899
10279	7	11039	29269	23	29622
11793	8	12540	29957	24	30274
13279	9	14010	30572	25	30853
14733	10	15447	31114	26	31357
16151	11	16846	31581	27	31786
17531	12	18205	31972	28	32138
18868	13	19520	32286	29	32413
20160	14	20788	32522	30	32610
21403	15	22006	32679	31	32729
22595			infinity		

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

**INDEX5 TABLE (as of 9-23-74)**

X(J)	J	R(J)	X(J)	J	R(J)
0	0	0	22006	8	23170
1608	1	3212	24279	9	25330
4808	2	6393	26320	10	27246
7962	3	9512	28106	11	28899
11039	4	12540	29622	12	30274
14010	5	15447	30853	13	31357
16846	6	18205	31786	14	32138
19520	7	20788	32413	15	32610
22006			infinity		

NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)

APPENDIX 2  
IMPLEMENTATION RECOMMENDATIONS

1. It is recommended that the priority-bit be turned ON in the HOST/IMP header.
2. It is recommended that in all abbreviations, "R" be used for Receiver and "X" for Transmitter.
3. The following identifiers and values are recommended for implementations:

SLNCTH 30 SILENCE-THRESHOLD.

Used for LONG-SILENCE definition. See below. Measured in the same units as GAIN, in its X-table.

TBS 1.000 sec TIME-BEGIN-SILENCE.

LONG-SILENCE is declared if GAIN<SLNCTH for more than TBS.

TAS 0.500 sec TIME-AFTER-SILENCE.

A delay introduced by the receiver after the end of LONG-SILENCE, before restarting the playback.

TES 0.150 sec TIME-END-SILENCE.

The amount of time the transmitter backs up at the end of a LONG-SILENCE in order to ensure a smooth transition back to speech.

TRI 2.000 sec TIME-RESPONSE-INITIAL.

Time for waiting for response for an initial call (#1 and #3). The initial call is repeated every TRI until an answer arrives, or until TRIGU expires.

TRIGU 20.000 sec TIME-RESPONSE-INITIAL-GIVEUP.

If no response to an initial call is received within TRIGU after the FIRST initial call, the system gives up, assuming the other system is down.

TRQ 1.000 sec TIME-RESPONSE-INQUIRY.

If no response to an inquiry (#8) is received within TRQ, the inquiry is repeated.

TRQGU 10.000 sec TIME-RESPONSE-INQUIRY-GIVEUP.

If no response to an inquiry is received within TRQGU from the FIRST inquiry, the system gives up, assuming the other system is down.

TBDA 3.000 sec TIME-BETWEEN-DATA-ARRIVAL.

If no data arrives within TBDA, an INQUIRY (#8) is sent. This repeats every TBDA.

TNR 2.000 sec TIME-NOT-READY.

**NETWORK VOICE PROTOCOL (NVP)**  
**RFC 741, NIC 42444 (Nov. 22, 1977)**

If the other system is in the NOT-READY (#7) state for more than TNR, an INQUIRY (#8) is sent. This repeats every TNR.

TNRGU 10.000 sec TIME-NOT-READY-GIVEUP.

If the other system is in the NOT-READY (#7) state for more than TNRGU, then the system gives up, assuming the other system is down.

TBIN 3.000 sec TIME-BUFFER-IN.

The input buffer size is equivalent to the time period TBIN (and its size is the DATA-RATE multiplied by the period TBIN). If the INPUT QUEUE ever gets to be longer than TBIN, data is discarded.

TBOUT 3.000 sec TIME-BUFFER-OUT.

The output buffer size is equivalent to the time period TBOUT (and its size is the DATA-RATE multiplied by the period TBOUT). If the OUTPUT QUEUE ever gets to be longer than TBOUT, data is discarded.

**NETWORK VOICE PROTOCOL (NVP)  
RFC 741, NIC 42444 (Nov. 22, 1977)**

**REFERENCES**

Bolt Beranek & Newman, Inc., Report No. 1822, Interface Message Processor: Specifications for the Interconnection of a Host and an IMP.

NSC Note 42 (in progress).

NSC Note 36, Proposal for NSC-LPC Coding/Decoding Tables, by J. D. Markel, Speech Communications Research Laboratory, Inc., July 20, 1974.

NSC Note 45, Everything You Always Wanted to Know about Gain, by E. Randolph Cole, USC/Information Sciences Institute, October 11, 1974.

NSC Note 56, Nothing to Lose, but Lots to Gain, by John Makhoul and Lynn Cosell, Bolt Beranek & Newman, Inc., March 10, 1975.

NSC Note 58, Gain Again, by Randy Cole, USC/Information Sciences Institute, March 12, 1975.



Protocol Specification  
NIC #7101

J. Postel  
UCLA - NMC  
Computer Science  
11 June 71

Official Initial Connection Protocol

DOCUMENT #2

This document specifies the third level protocol used to connect a user process at one site with a server process at another site. In one instance, the user process will be a Telnet and the server process will be a Logger.

This document describes a family of Initial Connection Protocols (ICP's) suitable for establishing one pair of connections between any user process and any server process. The description will be at two levels, the third or user level, and the second or NCP level.

Third Level Description

Notation

There is no standard notation for describing system calls which initiate and close connections or cause data to be sent, so the following *ad hoc* notation will be used.

Init (local =  $\ell$ , foreign =  $f$ , size =  $s$ )

causes the local Host to attempt to establish a connection between socket  $\ell$  at the local Host and socket  $f$ , with a byte size of  $s$  for the connection.

$\ell$  is a 32 bit local socket number.

$f$  is a 40 bit foreign socket number, the high-order eight bits of which specify the foreign Host, and

$s$  is an eight bit non-zero byte size.

The sum of  $\ell$  and  $f$  must be odd.

Listen (local =  $\ell$ , size =  $s$ )

causes the local Host to wait for a request for connection to local socket  $\ell$  with byte size  $s$ . The process will be awakened when a connection is established. The parameters  $\ell$  and  $s$  are the same as for Init.

Preceding page blank

Protocol Specification  
NIC #7101

J. Postel  
UCLA - NMC  
Computer Science  
11 June 71

Send (socket =  $\ell$ , data =  $d$ )

The data named by  $d$  is sent over the connection attached to local socket  $\ell$ .  $\ell$  must be a send socket attached to a connection.  $d$  is the name of a data area.

Receive (socket =  $\ell$ , data =  $d$ )

The receive side counterpart to send.

Close (socket =  $\ell$ )

Any connection currently attached to local socket  $\ell$  is closed.

#### A Family of ICP's

Briefly, a server process at a site attaches a well-advertised send socket L and listens. A user process initiates connection to L from its receive socket U. The byte size for this connection is 32. The server process then transmits a 32-bit even number S and closes the connection. The 32-bit number S and its successor, S+1, are the socket numbers the server will use. The final steps are for sockets S and S+1 at the server site to be connected to sockets U+3 and U+2 respectively at the user site.

Using the notation, the server executes the following sequence:

```
Listen (local = L, size = 32)
[Wait until a user connects]
Send (socket = L, data = S)
Close (socket = L)
Init (local = S, foreign = U+3, size = Bu)
Init (local = S+1, foreign = U+2, size = Bs)
```

The user executes the following:

```
Init (local = U, foreign = L, size = 32)
Receive (socket = U, data = S)
Optional Close (socket = U)
```

```
Listen (local = U+3, size = Bu)
or
Init (local = U+3, foreign = S, size = Bu)
Listen (local = U+2, size = Bs)
or
Init (local = U+2, foreign = S+1, size = Bs)
```

Note that L is a send socket (odd), while S and U are receive sockets (even). Where L, S or U are used as values of *local*, they are 32-bit numbers; where they are values of *foreign*, they are 40-bit numbers. The parameters  $B_s$  and  $B_u$  are the byte sizes to be sent by the server and user, respectively. If the user side declines to close socket U, then it must be handled automatically by the second level. (i.e. the NCP must send a matching CLS when it receives a CLS).

Examination of the above sequences reveals that an ICP is characterized by three numbers L,  $B_s$  and  $B_u$ , and must meet the restrictions that

- (a) L is a send socket,
- (b)  $B_s$  and  $B_u$  are legal byte sizes, and
- (c) for each L there is only one pair of associated byte sizes.

This last restriction prevents two distinct services from being available through the same socket and distinguished only by the byte sizes.

#### Second Level Description

##### Notation

The following notation will be used for the NCP Control Command used in ICP.

STR(ls, fs, s)

ls = local send socket  
fs = foreign receive socket  
s = byte size

Protocol Specification  
NIC #7101

J. Postel  
UCLA - NMC  
Computer Science  
11 June 71

RTS(ls, fs, l)

ls = local receive socket  
fs = foreign send socket  
l = link

ALL(l, m, b)

l = link  
m = message allocation  
b = bit allocation

CLS(ls, fs)

ls = local socket  
fs = foreign socket

The same family of ICP's is now described again.

Server

- S1: Listen on socket L.
- S2: Wait for a match.
- S3: STR(L, U, s<sub>1</sub>)
- S4: Wait for allocation.
- S5: Send data S in s<sub>1</sub> bit bytes as allowed by allocation m<sub>1</sub>, b<sub>1</sub>.
- S6: CLS(L, U)
- S7: RTS(S, U+3, l<sub>2</sub>)
- S8: STR(S+1, U+2, s<sub>3</sub>)

User

- U1: RTS(U, L, l<sub>1</sub>).
- U2: Wait for match.
- U3: ALL(l<sub>1</sub>, m<sub>1</sub>, b<sub>1</sub>)
- U4: Receive data S in s<sub>1</sub> bit bytes.
- U5: CLS(U, L)
- U6: STR(U+3, S, s<sub>2</sub>)
- U7: RTS(U+2, S+1, l<sub>3</sub>)

The labels here imply no ordering except that ordering required by the Host-Host Protocol. Note that steps S7 and S8 can be reversed as can U6 and U7. Also, notice that at any time after S2 the server could initiate steps S7 and S8 in parallel with steps S3 through S6, and that at any time after U4 the user could initiate steps U6 and U7 in parallel with step U5.

Protocol Specification  
NIC #7101

J. Postel  
UCLA - NMC  
Computer Science  
11 June 71

Following the above exchanges ALL commands would be exchanged and data transfers could begin.

At this level the parameters of the above ICP family are  $L$ ,  $m_1$ ,  $b_1$ ,  $s_1$   $s_2$   $s_3$   $\ell_1$   $\ell_2$   $\ell_3$ .

$L$  is a well known socket number and will be specified for each type of service.

$m_1$  and  $b_1$  are allocation quantities for the transfer of a socket number.

$m_1$  is specified to be at least 1.

$b_1$  is specified to be at least 32.

$s_1$ ,  $s_2$  and  $s_3$  are byte sizes. Only  $s_1$  is to be specified as  $s_2$  and  $s_3$  are to be left to the process involved.

$s_1$  is specified to be 32.

$\ell_1$ ,  $\ell_2$ , and  $\ell_3$  are links and are not specified.

It is legal for the NCP to receive RTS or STR before the corresponding local Init or Listen is issued.



7155

Network Working Group  
 Request for Comments: #202  
 NIC #7155  
 Categories: D  
 References: Document #2  
 Obsoletes: None

Steve Wolfe  
 UCLA-CCN  
 Jon Postel  
 UCLA-NMC  
 26 July 71

We have noticed a possible deadlock situation which may arise using the Initial Connection Protocol (ICP) specified in Document #2 (NIC #7101 in the Current Network Protocols Notebook NIC #7104).

If on both sides one RFC is issued and a "wait for match" is required before the second RFC is issued, it is possible that the first RFC's will not match. In particular a deadlock will occur if both sides open their send or both sides open their receive sockets first.

Briefly the ICP is:

Server

- S1: Listen on socket L.
- S2: Wait for a match
- S3: STR(L, U,  $\underline{s}_1$ )
- S4: Wait for allocation.
- S5: Send data S in  $\underline{s}_1$  bit bytes as allowed by allocation  $\underline{m}_1$ ,  $\underline{b}_1$ .
- S6: CLS(L, U)
- S7: RTS(S, U+3,  $\underline{\ell}_2$ )
- S8: STR(S+1, U+2,  $\underline{s}_3$ )

User

- U1: RTS(U, L,  $\underline{\ell}_1$ ).
- U2: Wait for match.
- U3: ALL( $\underline{\ell}_1$ ,  $\underline{m}_1$ ,  $\underline{b}_1$ )
- U4: Receive data S in  $\underline{s}_1$  bit bytes
- U5: CLS(U, L)
- U6: STR(U+3, S,  $\underline{s}_2$ )
- U7: RTS(U+2, S+1,  $\underline{\ell}_3$ )

"The labels here imply no ordering except that ordering required by the Host-Host Protocol. Note that steps S7 and S8 can be reversed as can U6 and U7. Also, notice that at any time after S2 the server could initiate steps S7 and S8 in parallel with steps S3 through S6, and that at any time after U4 the user could initiate steps U6 and U7 in parallel with step U5."

We recommend that the server perform steps 7 and 8 before waiting for the user to perform step 6 or 7. It is also suggested that the user issue the RFC's in steps 6 and 7 without waiting for the server. (If the user is only Listening then both Listens should be issued without waiting for the server.) If for some reason a host must delay between issuing RFC's it must issue the RFC's involving sockets S and U+3 first.



Protocol Specification  
NIC #7103

J. Postel  
UCLA - NMC  
Computer Science  
15 June 71

Official Telnet-Logger Initial Connection Protocol

DOCUMENT #3

For connecting Telnet and Logger processes, the ICP parameters are L=1,  $B_u = \underline{s}_2 = 8$ , and  $B_s = \underline{s}_3 = 8$ . (To clarify the socket number required, L = X'00000001').

---

Preceding page blank



TELNET PROTOCOL SPECIFICATION  
RFC 542, NIC 18639 (Aug. 1973)

TELNET PROTOCOL SPECIFICATION.

INTRODUCTION

The purpose of the TELNET Protocol is to provide a fairly general, bi-directional, eight-bit byte oriented communications facility. Its primary goal is to allow a standard method of interfacing terminal devices and terminal-oriented processes to each other. It is envisioned that the protocol may also be used for terminal-terminal communication ("linking") and process-process communication (distributed computation).

GENERAL CONSIDERATIONS

A TELNET connection consists of a pair of standard Host/Host Protocol connections over which passes data with interspersed TELNET control information. The pair of connections are typically established by the Initial Connection Protocol. Details on the Host/Host and Initial Connection Protocols may be found elsewhere in NIC #7104.

The TELNET Protocol is built upon three main ideas: first, the concept of a "Network Virtual Terminal"; second, the principle of negotiated options; and third, a symmetric view of terminals and processes.

1. When a TELNET connection is first established, each end is assumed to originate and terminate at a "Network Virtual Terminal", or NVT. An NVT is an imaginary device which provides a standard, network-wide, intermediate representation of a canonical terminal. This eliminates the need for "server" and "user" Hosts\* to keep information about the characteristics of each other's terminals and terminal handling conventions. All Hosts, both user and server, map their local device characteristics and conventions so as to appear to be dealing with an NVT over the network, and each can assume a similar mapping by the other party. The NVT is intended to strike a balance between being overly restricted (not providing Hosts a rich enough vocabulary for mapping into their local character sets), and being overly inclusive (penalizing users with modest terminals).

\*NOTE: The "user" Host is the Host to which the physical terminal is normally attached, and the "server" host is the Host which is normally providing some service. As an alternate point of view, applicable even in terminal-to-terminal or process-to-process communications, the "user" Host is the Host which initiated the communication.

2. The principle of negotiated options takes cognizance of the fact that many sites will wish to provide additional services over and above those available within an NVT, and many users will have sophisticated terminals and would like to have elegant, rather than minimal, services. Independent of, but structured within, the TELNET Protocol various "options" will be sanctioned which can be used with the "DO, DON'T, WILL, WON'T" structure (discussed below) to allow a user and server to agree to use a more elaborate (or perhaps just different) set of conventions for their TELNET connection. Such options could include changing the character set, the echo mode, the line width, the page length, etc.

The basic strategy for setting up the use of options is to have either party (or both) initiate a request that some option take effect. The other party may then either accept or reject the request. If the request is accepted the option immediately takes effect; if it is rejected the

**TELNET PROTOCOL SPECIFICATION**  
**RFC 542, NIC 18639 (Aug. 1973)**

associated aspect of the connection remains as specified for an NVT. Clearly, a party may always refuse a request to enable, and must never refuse a request to disable, some option since all parties must be prepared to support the NVT.

The syntax of option negotiation has been set up so that if both parties request an option simultaneously, each will see the other's request as the positive acknowledgment of its own.

3. The symmetry of the negotiation syntax can potentially lead to nonterminating acknowledgment loops -- each party seeing the incoming commands not as acknowledgments but as new requests which must be acknowledged. To prevent such loops, the following rules prevail:

- a. Parties may only request a change in option status; i.e., a party may not send out a "request" merely to announce what mode it is in.
- b. If a party receives what appears to be a request to enter some mode it is already in, the request should not be acknowledged.
- c. Whenever one party sends an option command to a second party, whether as a request or an acknowledgment, and use of the option will have any effect on the processing of the data being sent from the first party to the second, then the command must be inserted in the data stream at the point where it is desired that it take effect. (It should be noted that some time will elapse between the transmission of a request and the receipt of an acknowledgment, which may be negative. Thus, a site may wish to buffer data, after requesting an option, until it learns whether the request is accepted or rejected, in order to hide the "uncertainty period" from the user.)

Option requests are likely to flurry back and forth when a TELNET connection is first established, as each party attempts to get the best possible service from the other party. Beyond that, however, options can be used to dynamically modify the characteristics of the connection to suit changing local conditions. For example, the NVT, as will be explained later, uses a transmission discipline well suited to the many "line at a time" applications such as BASIC, but poorly suited to the many "character at a time" applications such as NLS. A server might elect to devote the extra processor overhead required for a "character at a time" discipline when it was suitable for the local process and would negotiate an appropriate option. However, rather than then being permanently burdened with the extra processing overhead, it could switch (i.e., negotiate) back to NVT when the more "taut" control was no longer necessary.

It is possible for requests initiated by processes to simulate a nonterminating request loop if the process responds to a rejection by merely re-requesting the option. To prevent such loops from occurring, rejected requests should not be repeated until something changes. Operationally, this can mean the process is running a different program, or the user has given another command, or whatever makes sense in the context of the given process and the given option. A good rule of thumb is that a re-request should only occur as a result of subsequent information from the other end of the connection or when demanded by local human intervention.

Option designers should not feel constrained by the somewhat limited syntax available for option negotiation. The intent of the simple syntax is to make it easy to have options--since it is correspondingly easy to profess ignorance about them. If some particular option requires

**TELNET PROTOCOL SPECIFICATION**  
**RFC 542, NIC 18639 (Aug. 1973)**

a richer negotiation structure than possible within "DO, DON'T, WILL, WON'T", the proper tack is to use "DO, DON'T, WILL, WON'T" to establish that both parties understand the option, and once this is accomplished a more exotic syntax can be used freely. For example, a party might send a request to alter (establish) line length. If it is accepted, then a different syntax can be used for actually negotiating the line length--such a "sub-negotiation" perhaps including fields for minimum allowable, maximum allowable and desired line lengths. The important concept is that such expanded negotiations should never begin until some prior (standard) negotiation has established that both parties are capable of parsing the expanded syntax.

In summary, WILL XXX is sent, by either party, to indicate that party's desire (offer) to begin performing option XXX, DO XXX and DON'T XXX being its positive and negative acknowledgments; similarly, DO XXX is sent to indicate a desire (request) that the other party (i.e., the recipient of the DO) begin performing option XXX, WILL XXX and WON'T XXX being the positive and negative acknowledgments. Since the NVT is what is left when no options are enabled, the DON'T and WON'T responses are guaranteed to leave the connection in a state which both ends can handle. Thus, all Hosts may implement their TELNET processes to be totally unaware of options that are not supported, simply returning a rejection to (i.e., refusing) any option request that cannot be understood.

As much as possible, the TELNET protocol has been made server-user symmetrical so that it easily and naturally covers the user-user (linking) and server-server (cooperating processes) cases. It is hoped, but not absolutely required, that options will further this intent. In any case, it is explicitly acknowledged that symmetry is an operating principle rather than an ironclad rule.

A companion document, "TELNET Option Specifications," should be consulted for information about the procedure for establishing new options. That document, as well as descriptions of all currently defined options, is contained in the TELNET section of the ARPANET Protocol Handbook (NIC #7104).

#### THE NETWORK VIRTUAL TERMINAL

The Network Virtual Terminal (NVT) is a bi-directional character device. The NVT has a printer and a keyboard. The printer responds to incoming data and the keyboard produces outgoing data which is sent over the TELNET connection and, if "echoes" are desired, to the NVT's printer as well. "Echoes" will not be expected to traverse the network (although options exist to enable a "remote" echoing mode of operation, no Host is required to implement this option). The code set is seven-bit USASCII in an eight-bit field), except as modified herein. Any code conversion and timing considerations are local problems and do not affect the NVT.

#### TRANSMISSION OF DATA

Although a TELNET connection through the network is intrinsically full duplex, the NVT is to be viewed as a half-duplex device operating in a line-buffered mode. That is, unless and until options are negotiated to the contrary, the following default conditions pertain to the transmission of data over the TELNET connection:

**TELNET PROTOCOL SPECIFICATION**  
**RFC 542, NIC 18639 (Aug. 1973)**

- 1) Insofar as the availability of local buffer space permits, data should be accumulated in the Host where it is generated until a complete line of data is ready for transmission, or until some locally-defined explicit signal to transmit occurs. This signal could be generated either by a process or by a human user.

The motivation for this rule is the high cost, to some Hosts, of processing network input interrupts, coupled with the default NVT specification that "echoes" do not traverse the network. Thus, it is reasonable to buffer some amount of data at its source. Many systems take some processing action at the end of each input line (even line printers or card punches frequently tend to work this way), so the transmission should be triggered at the end of a line. On the other hand, a user or process may sometimes find it necessary or desirable to provide data which does not terminate at the end of a line; therefore implementers are cautioned to provide methods of locally signaling that all buffered data should be transmitted immediately.

- 2) When a process has completed sending data to an NVT printer and has no queued input from the NVT keyboard for further processing (i.e., when a process at one end of a TELNET connection cannot proceed without input from the other end), the process must transmit the TELNET Go Ahead (GA) command.

This rule is not intended to require that the TELNET GA command be sent from a terminal at the end of each line, since server Hosts do not normally require a special signal (in addition to end-of-line or other locally-defined characters) in order to commence processing. Rather, the TELNET GA is designed to help a user's local Host operate a physically half duplex terminal which has a "lockable" keyboard such as the IBM 2741. A description of this type of terminal may help to explain the proper use of the GA command.

The terminal-computer connection is always under control of either the user or the computer. Neither can unilaterally seize control from the other; rather the controlling end must relinquish its control explicitly. At the terminal end, the hardware is constructed so as to relinquish control each time that a "line" is terminated (i.e., when the "New Line" key is typed by the user). When this occurs, the attached (local) computer processes the input data, decides if output should be generated, and if not returns control to the terminal. If output should be generated, control is retained by the computer until all output has been transmitted.

The difficulties of using this type of terminal through the network should be obvious. The "local" computer is no longer able to decide whether to retain control after seeing an end-of-line signal or not; this decision can only be made by the "remote" computer which is processing the data. Therefore, the TELNET GA command provides a mechanism whereby the "remote" (server) computer can signal the "local" (user) computer that it is time to pass control to the user of the terminal. It should be transmitted at those times, and only at those times, when the user should be given control of the terminal. Note that premature transmission of the GA command may result in the blocking of output, since the user is likely to assume that the transmitting system has paused, and therefore he will fail to turn the line around manually.

The foregoing, of course, does not apply to the user-to-server direction of communication. In this direction, GAs may be sent at any time, but need not ever be sent. Also, if the TELNET connection is being used for process-to-process communication, GAs need not be

TELNET PROTOCOL SPECIFICATION  
RFC 542, NIC 18639 (Aug. 1973)

sent in either direction. Finally, for terminal-to-terminal communication, GAs may be required in neither, one, or both directions. If a Host plans to support terminal-to-terminal communication it is suggested that the Host provide the user with a means of manually signaling that it is time for a GA to be sent over the TELNET connection; this, however, is not a requirement on the implementer of a TELNET process.

#### STANDARD REPRESENTATION OF CONTROL FUNCTIONS

As stated in the Introduction to this document, the primary goal of the TELNET protocol is the provision of a standard interfacing of terminal devices and terminal-oriented processes through the network. Early experiences with this type of interconnection have shown that certain functions are implemented by most servers, but that the methods of invoking these functions differ widely. For a human user who interacts with several server systems, these differences are highly frustrating. TELNET, therefore, defines a standard representation for five of these functions, as described below. These standard representations have standard, but not required, meanings (with the exception that the IP function may be required by other protocols which use TELNET); that is, a system which does not provide the function to local users need not provide it to network users and may treat the standard representation for the function as a No-operation. On the other hand, a system which does provide the function to local is obliged to provide the same function as a network user who transmits the standard representation for the function.

##### Interrupt Process (IP)

Many systems provide a function which suspends, interrupts, aborts, or terminates the operation of a user process. This function is frequently used when a user believes his process is in an unending loop, or when an unwanted process has been inadvertently activated. IP is the standard representation for invoking this function. It should be noted by implementers that IP may be required by other protocols which use TELNET, and therefore should be implemented if these other protocols are to be supported.

##### Abort Output (AO)

Many systems provide a function which allows a process, which is generating output, to run to completion (or to reach the same stopping point it would reach if running to completion) but without sending the output to the user's terminal. Further, this function typically clears any output already produced but not yet actually printed (or displayed) on the user's terminal. AO is the standard representation for invoking this function. For example, some subsystem might normally accept a user's command, send a long text string to the user's terminal in response, and finally signal readiness to accept the next command by sending a "prompt" character (preceded by <CR><LF>) to the user's terminal. If the AO were received during the transmission of the text string, a reasonable implementation would be to suppress the remainder of the text string, but transmit the prompt character and the preceding <CR><LF>. (This is possibly in distinction to the action which might be taken if an IP were received; the IP might cause suppression of the text string and an exit from the subsystem.)

It should be noted, by systems which provide this function, that there may be buffers external to the system (in the network and the user's "local" Host) which should be cleared; the appropriate way to do this is to transmit the "Synch" signal described below.

**TELNET PROTOCOL SPECIFICATION**  
**RFC 542, NIC 18639 (Aug. 1973)**

**Are You There (AYT)**

Many systems provide a function which provides the user with some visible (e.g., printable) evidence that the system is still up and running. This function may be invoked by the user when the system is unexpectedly "silent" for a long time, because of the unanticipated (by the user) length of a computation, an unusually heavy system load, etc. AYT is the standard representation for invoking this function.

**Erase Character (EC)**

Many systems provide a function which deletes the last preceding undeleted character or "print position" \* from the stream of data being supplied by the user. This function is typically used to edit keyboard input when typing mistakes are made. EC is the standard representation for invoking this function.

\*NOTE: A "print position" may contain several characters which are the result of overstrikes, or of sequences such as <char1> BS <char2>...

**Erase Line (EL)**

Many systems provide a function which deletes all the data in the current "line" of input. This function is typically used to edit keyboard input. EL is the standard representation for invoking this function.

**THE TELNET "SYNCH" SIGNAL**

Most time-sharing systems provide mechanisms which allow a terminal user to regain control of a "runaway" process; the IP and AO functions described above are examples of these mechanisms. Such systems, when used locally, have access to all of the signals supplied by the user, whether these are normal characters or special "out of band" signals such as those supplied by the teletype "BREAK" key or the IBM 2741 "ATTN" key. This is not necessarily true when terminals are connected to the system through the network; the network's flow control mechanisms may cause such a signal to be buffered elsewhere, for example in the user's Host.

To counter this problem, the TELNET "Synch" mechanism is introduced. A Synch signal consists of a Host/Host Protocol INS command, coupled with the TELNET command DATA MARK. The INS command, which is not subject to the flow control pertaining to the TELNET connections, is used to invoke special handling of the data stream by the process which receives it. In this mode, the data stream is immediately scanned for "interesting" signals as defined below, discarding intervening data. The TELNET command DATA MARK (DM) is the synchronizing mark in the data stream which indicates that any special signal has already occurred and the recipient can return to normal processing of the data stream. When a DM arrives before its associated INS, the recipient should not process the data stream further until the matching INS is received, in order to insure that the two ends of the connection remain synchronized. Also, implementers are warned that in some cases several Synch's may be sent in succession. In general, this will require a count of the INSs received so as to properly pair them with the associated DM's. "Interesting" signals are defined to be: the TELNET standard representations of IP, AO, and AYT (but not EC

TELNET PROTOCOL SPECIFICATION  
RFC 542, NIC 18639 (Aug. 1973)

or EL); the local analogs of these standard representations (if any); all other TELNET commands; other site-defined signals which can be acted on without delaying the scan of the data stream.

Since one effect of the SYNCH mechanism is the discarding of essentially all characters (except TELNET commands) between the sender of the Synch and its recipient, this mechanism is specified as the standard way to clear the data path when that is desired. For example, if a user at a terminal causes an AO to be transmitted, the server which receives the AO (if it provides that function at all) should return a Synch to the user.

Finally, just as the NCP-level INS command is needed at the TELNET level as an out-of-band signal, so other protocols which make use of TELNET may require a TELNET command which can be viewed as an out-of-band signal at a different level.

By convention the sequence [IP, Synch] is to be used as such a signal. For example, suppose that some other protocol, which uses TELNET, defines the character string STOP analogously to the TELNET command AO. Imagine that a user of this protocol wishes a server to process the STOP string, but the connection is blocked because the server is processing other commands. The user should instruct his system to:

1. Send the TELNET IP character;
2. Send the TELNET SYBC sequence, that is:
  - a. Send the TELNET Data Mark (DM);
  - b. Send the Host-Host Protocol INS;
3. Send the character string STOP; and
4. Send the other protocol's analog of the TELNET DM, if any.

The user (or process acting on his behalf) must transmit the TELNET SYNCH sequence of step 2 above to ensure that the TELNET IP gets through to the server's TELNET interpreter.

#### THE NVT PRINTER AND KEYBOARD

The NVT printer has an unspecified carriage width and page length and can produce representations of all 95 USASCII graphics (codes 32 through 126). Of the 33 USASCII control codes (0 through 31 and 127), and the 128 uncovered codes (128 through 255), the following have specified meaning to the NVT printer:

NAME	CODE	MEANING
NULL (NUL) Line Feed (LF)	0 10	A no operation Moves the printer to the next print line, keeping the same horizontal position
Carriage Return (CR)	13	Moves the printer to the left margin of the current line.

**TELNET PROTOCOL SPECIFICATION**  
**RFC 542, NIC 18639 (Aug. 1973)**

In addition, the following codes shall have defined, but not required, effects on the NVT printer. Neither end of a TELNET connection may assume that the other party will take, or will have taken, any particular action upon receipt or transmission of these:

BELL (BEL)	7	Produces an audible or visible signal (which does NOT move the print head)
Back Space (BS)	8	Moves the print head one character position towards the left margin.
Horizontal Tab (HT)	9	Moves the printer to the next horizontal tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
Vertical Tab (VT)	11	Moves the printer to the next horizontal tab stop. It remains unspecified how either party determines or establishes where such tab stops are located.
Form Feed (FF)	12	Moves the printer to the top of the next page, keeping the same horizontal position

All remaining codes do not cause the NVT printer to take any action.

The sequence "CR LF", as defined, will cause the NVT to be positioned at the left margin of the next print line (as would, for example, the sequence "LF CR"). However, many systems and terminals do not treat CR and LF independently, and will have to go to some effort to simulate their effect. (For example, some terminals do not have a CR independent of the LF, but on such terminals it may be possible to simulate a CR by backspacing.) Therefore, the sequence "CR LF" must be treated as a single "new line" character and used whenever their combined action is intended; the sequence "CR NUL" must be used where a carriage return alone is actually desired; and the CR character must be avoided in other contexts. This rule gives assurance to systems which must decide whether to perform a "new line" function or a multiple-backspace that the TELNET stream contains a character following a CR that will allow a rational decision.

The NVT keyboard has keys, or key combinations, or key sequences, for generating all 128 USASCII codes. Note that although many have no effect on the NVT printer, the NVT keyboard is capable of generating them.

In addition to these codes, the NVT keyboard shall be capable of generating the following additional codes which, except as noted, have defined, but not required, meanings. The actual code assignments for these "characters" are in the TELNET Command section, because they are viewed as being, in some sense, generic and should be available even when the data stream is interpreted as being some other character set.

Synch

This key allows the user to clear his data path to the other party. The activation of

TELNET PROTOCOL SPECIFICATION  
RFC 542, NIC 18639 (Aug. 1973)

this key causes a DM (see command section) to be sent in the data stream and an INS to be sent on the control link. The pair DM-INS is to have required meaning as defined previously.

Break (BRK)

This code is provided because it is a signal outside the USASCII set which is currently given local meaning within many systems. It is intended to indicate that the Break Key or the Attention Key was hit. Note, however, that this is intended to provide a 129th code for systems which require it, not as a synonym for the IP standard representation.

Interrupt Process (IP)

Suspend, interrupt, abort or terminate the process to which the NVT is connected. Also, part of the out-of-band signal for other protocols which use TELNET.

Abort Output (AO)

Allow the current process to (appear to) run to completion, but do not send its output to the user. Also, send a Synch to the user.

Are You There (AYT)

Send back to the NVT some visible (i.e., printable) evidence that the AYT was received.

Erase Character (EC)

The recipient should delete the last preceding undeleted character or "print position" from the data stream.

Erase Line (EL)

The recipient should delete characters from the data stream back to, but not including, the last "CR LF" sequence sent over the TELNET connection.

The spirit of these "extra" keys, and also the printer format effectors, is that they should represent a natural extension of the mapping that already must be done from "NVT" into "local". Just as the NVT data byte 104 should be mapped into whatever the local code for "uppercase D" is, so the EC character should be mapped into whatever the local "Erase Character" function is. Further, just as the mapping for 174 is somewhat arbitrary in an environment that has no "vertical bar" character, the EL character may have a somewhat arbitrary mapping (or none at all) if there is no local "Erase Line" facility. Similarly for format effectors: if the terminal actually does have a "Vertical tab", then the mapping for VT is obvious, and only when the terminal does not have a vertical tab should the effect of VT be unpredictable.

TELNET COMMAND STRUCTURE

All TELNET commands consist of at least a two byte sequence: the "Interpret as Command" (IAC) escape character followed by the code for the command. The commands dealing with option negotiation are three byte sequences, the third byte being the code for the option referenced. This format was chosen so that as more comprehensive use of the "data space"

**TELNET PROTOCOL SPECIFICATION**  
**RFC 542, NIC 18639 (Aug. 1973)**

is made -- by negotiations from the basic NVT, of course -- collisions of data bytes with reserved command values will be minimized, all such collisions requiring the inconvenience, and inefficiency, of "escaping" the data bytes into the stream. With the current set-up, only the IAC need be doubled to be sent as data, and the other 255 codes may be passed transparently.

The following are the defined TELNET commands. Note that these codes and code sequences have the indicated meaning only when immediately preceded by an IAC.

NAME	CODE	MEANING
SE	240	End of subnegotiation parameters
NOP	241	No operation
Data Mark	242	The data stream portion of a Synch This should always be accompanied by an INS on the control link.
Break	243	NVT character BRK
Interrupt Process	244	The function IP
Abort output	245	The function AO
Are You There	246	The function AYT
Erase character	247	The function EC
Erase Line	248	The function EL
Go ahead	249	The GA signal
SB	250	Indicates that what follows is subnegotiation of the indicated option
WILL (option code)	251	Indicates the desire to begin performing, or confirmation that you are now performing, the indicated option
WON'T (option code)	252	Indicates the refusal to perform, or continue performing, the indicated option.
DO (option code)	253	Indicates the request that the other party perform, or confirmation that you are expecting the other party to perform, the indicated option.
DON'T (option code)	254	Indicates the demand that the other party stop performing, or confirmation that you are no longer expecting the other party to perform, the indicated option.
IAC	255	Data Byte 255.

**TELNET OPTION SPECIFICATIONS**  
**NIC 18640 (Aug. 1973)**

**TELNET OPTION SPECIFICATIONS**

The intent of providing for options in the TELNET Protocol is to permit sites to obtain more elegant solutions to the problems of communication between dissimilar devices than is possible within the framework provided by the Network Virtual Terminal (NVT). It should be possible for sites to invent, test, or discard options at will, even though the negotiation method permits the direct use of only 256 option codes. Nevertheless, it is envisioned that options which prove to be generally useful will eventually be supported by many sites; therefore it is desirable that options should be carefully documented and well publicized. In addition, it is necessary to insure that a single option code is not used for several different options.

This document specifies a method of option code assignment and standards for documentation of options. The individual responsible for assignment of option codes may waive the requirement for complete documentation for some cases of experimentation, but in general documentation will be required prior to code assignment. Options will be publicized by including their documentation in the TELNET section of the ARPANET Protocol Handbook (NIC #7104); inventors of options may, of course, publicize them in other ways as well.

Option codes will be assigned by:

Jonathan B. Postel  
University of Southern California  
Information Sciences Institute (USC-ISI)  
4676 Admiralty Way  
Marina Del Rey, California 90291  
(213) 822-1511

SNDMSG mailbox = POSTEL@USC-ISIB  
NLS Sendmail Ident = JBP

Documentation of options should contain at least the following sections:

Section 1 – Command name (and option code).

Section 2 – Command meanings (definitions).

The meaning of each possible TELNET command relevant to this option should be described. Note that for complex options, where "subnegotiation" is required, there may be a large number of possible commands. The concept of "subnegotiation" is described in more detail below.

Section 3 – Default specification.

The default assumptions for sites which do not choose to implement, or use, the option must be described.

Section 4 – Motivation.

A detailed explanation of the motivation for inventing a particular option, or for choosing a particular form for the option, is extremely helpful to others who are not faced (or don't realize that they are faced) by the problem that the option is designed to solve.

**TELNET OPTION SPECIFICATIONS**  
**NIC 18640 (Aug. 1973)**

**Section 5 – Description (or Implementation Rules).**

Merely defining the command meanings and providing a statement of motivation are not always sufficient to insure that two implementations of an option will be able to communicate. Therefore, a more complete description should be furnished in most cases. This description might take the form of text, a sample implementation, hints to implementers, etc.

**A Note on "Subnegotiation"**

Some options will require more information to be passed between sites than a single option code. For example, any option which requires a parameter is such a case. The strategy to be used consists of two steps: first, both parties agree to "discuss" the parameter(s) and, second, the "discussion" takes place.

The first step, agreeing to discuss the parameters, takes place in the normal manner; one party proposes use of the option by sending a DO (or WILL) followed by the option code, and the other party accepts by returning a WILL (or DO) followed by the option code. Once both parties have agreed to use the option, subnegotiation takes place by using the command SB, followed by the option code, followed by the parameter(s), followed by the command SE. Each party is presumed to be able to parse the parameter(s), since each has indicated that the option is supported (via the initial exchange of WILL and DO). On the other hand, the receiver may locate the end of a parameter string by searching for the SE command (i.e., the string IAC SE), even if the receiver is unable to parse the parameters. Of course, either party may refuse to pursue further subnegotiation at any time by sending a WON'T or DON'T to the other party.

Thus, for option "ABC", which requires subnegotiation, the formats of the TELNET commands are:

IAC WILL ABC

Offer to use option ABC (or favorable acknowledgment of other party's request)

IAC DO ABC

Request for other party to use option ABC (or favorable acknowledgment of other party's offer)

IAC SB ABC <parameters> IAC SE

One step of subnegotiation, used by either party.

Designers of options requiring "subnegotiation" must take great care to avoid unending loops in the subnegotiation process. For example, if each party can accept any value of a parameter, and both parties suggest parameters with different values, then one is likely to have an infinite oscillation of "acknowledgments" (where each receiver believes it is only acknowledging the new proposals of the other). Finally, if parameters in an option "subnegotiation" include a byte with a value of 255, it is necessary to double this byte in accordance the general TELNET rules.

Telnet Option Index  
NIC 29610 (Dec. 9, 1977)

TELNET Option Index

NUMBER	NAME	RFC	NIC
0	Binary Transmission	...	15389
1	Echo	...	15390
2	Reconnection	...	15391
3	Suppress Go Ahead	...	15392
4	Approximate Message Size Negotiation	...	15393
5	Status	651	31154
6	Timing Mark	...	16238
7	Remote Controlled Transmission and Echoing	726	39237
8	Output Line Width	...	20196
9	Output Page Size	...	20197
10	Output Carriage-Return Disposition	652	31155
11	Output Horizontal Tabstops	653	31156
12	Output Horizontal Tab Disposition	654	31157
13	Output Formfeed Disposition	655	31158
14	Output Vertical Tabstops	656	31159
15	Output Vertical Tab Disposition	657	31160
16	Output Linefeed Disposition	658	31161
17	Extended ASCII	698	32964
18	Logout	727	40025
19	Byte Macro	735	42083
20	Data Entry Terminal	732	41762
21	SUPDUP	736	42213
21	SUPDUP Protocol	734	41953
255	Extended-Options-List	...	16239



TELNET BINARY TRANSMISSION OPTION  
NIC 15389 (Aug. 1973)

TELNET BINARY TRANSMISSION OPTION

1. Command name and code.

TRANSMIT-BINARY 0

2. Command meanings.

IAC WILL TRANSMIT-BINARY

The sender of this command REQUESTS permission to begin transmitting, or confirms that it will now begin transmitting characters which are to be interpreted as 8 bits of binary data by the receiver of the data.

IAC WON'T TRANSMIT-BINARY

If the connection is already being operated in binary transmission mode,\* the sender of this command DEMANDS to begin transmitting data characters which are to be interpreted as standard NVT ASCII characters by the receiver of the data. If the connection is not already being operated in binary transmission mode, the sender of this command REFUSES to begin transmitting characters which are to be interpreted as binary characters by the receiver of the data (i.e., the sender of the data demands to continue transmitting characters in its present mode).

IAC DO TRANSMIT-BINARY

The sender of this command REQUESTS that the sender of the data start transmitting, or confirms that the sender of data is expected to transmit, characters which are to be interpreted as 8 bits of binary data (i.e., by the party sending this command.)

IAC DON'T TRANSMIT-BINARY

If the connection is already being operated in binary transmission mode,\* the sender of this command DEMANDS that the sender of the data start transmitting characters which are to be interpreted as standard NVT ASCII characters by the receiver of the data (i.e., the party sending this command). If the connection is not already being operated in binary transmission mode, the sender of this command DEMANDS that the sender of data continue transmitting characters which are to be interpreted in the present mode.

\*A connection is being operated in binary transmission mode only when one party has requested it and the other has acknowledged it.

Preceding page blank

TELNET BINARY TRANSMISSION OPTION  
NIC 15389 (Aug. 1973)

3. Default

WON'T TRANSMIT-BINARY

DON'T TRANSMIT-BINARY

i.e., won't switch to binary mode (if not already in it) or switching back to NVT ASCII mode (if presently in binary mode).

4. Motivation for the option.

It is sometimes useful to have available a binary transmission path within TELNET without having to utilize one of the more efficient, higher level protocols providing binary transmission (such as the File Transfer Protocol). The use of the IAC prefix within the basic TELNET protocol provides the option of binary transmission in a natural way, requiring only the addition of a mechanism by which the parties involved can agree to INTERPRET the characters transmitted over a TELNET connection as binary data.

5. Description of the option.

With the binary transmission option in effect, the receiver should interpret characters received from the transmitter which are not preceded with IAC as 8 bit binary data, with the exception of IAC followed by IAC which stands for the 8 bit binary data with the decimal value 255. IAC followed by an effective TELNET command (plus any additional characters required to complete the command) is still the command even with the binary transmission option in effect. IAC followed by a character which is not a defined TELNET command has the same meaning as IAC followed by NOP, although an IAC followed by an undefined command should not normally be sent in this mode.

6. Implementation suggestions.

It is foreseen that implementations of the binary transmission option will choose to refuse some other options (such as the EBCDIC transmission option) while the binary transmission option is in effect. However, if a pair of Hosts can understand being in binary transmission mode simultaneous with being in, for example, echo mode, then it is all right if they negotiate that combination. Some options in combination with the binary transmission option look very useful, such as negotiate line length (i.e., buffer length).

It should be mentioned that the meanings of WON'T and DON'T are dependent upon whether the connection is presently being operated in binary mode or not. Consider a connection operating in, say, EBCDIC

TELNET BINARY TRANSMISSION OPTION  
NIC 15389 (Aug. 1973)

mode which involves a system which has chosen not to implement any knowledge of the binary command. If this system were to receive a DO TRANSMIT-BINARY, it would not recognize the TRANSMIT-BINARY option and therefore would return a WON'T TRANSMIT-BINARY. If the default for the WON'T TRANSMIT-BINARY were always NVT ASCII, the sender of the DO TRANSMIT-BINARY would expect the recipient to have switched to NVT ASCII, whereas the receiver of the DO TRANSMIT-BINARY would not make this interpretation.

Thus, we have the rule that when a connection is not presently operating in binary mode, the default (i.e., the interpretation of WON'T and DON'T) is to continue operating in the current mode, whether that is NVT ASCII, EBCDIC, or some other mode. This rule, however, is not applied once a connection is operating in a binary mode (as agreed to by both ends); this would require each end of the connection to maintain a stack, containing all of the encoding-method transitions which had previously occurred on the connection, in order to properly interpret a WON'T or DON'T. Thus, a WON'T or DON'T received after the connection is operating in binary mode causes the encoding method to revert to NVT ASCII.

It should be remembered that a TELNET connection is normally a PAIR of connections, one going each way; operating in binary transmission mode must be negotiated separately for each of the pair of connections, if that is desired.

Implementation of the binary transmission option, as is the case with implementations of all other TELNET options, must follow the loop preventing rules given in the General Considerations section of the TELNET Protocol Specification.

Consider now some issues of binary transmission both to and from both a process and a terminal:

a. Binary transmission from a terminal.

The implementer of the binary transmission option should consider how (or whether) a terminal transmitting over a TELNET connection with binary transmission in effect is allowed to generate all eight bit characters, ignoring parity considerations, etc., on input from the terminal.

TELNET BINARY TRANSMISSION OPTION  
NIC 15389 (Aug. 1973)

b. Binary transmission to a process.

The implementer of the binary transmission option should consider how (or whether) all characters are passed to a process receiving over a connection with binary transmission in effect. As an example of the possible problem, TENEX intercepts certain characters (e.g., ETX, the Teletype control-C) at monitor level and does not pass them to the process.

c. Binary transmission from a process.

The implementer of the binary transmission option should consider how (or whether) a process transmitting over a connection with binary transmission in effect is allowed to send all eight bit characters with no characters intercepted by the monitor and changed to other characters. An example of such a conversion may be found in the TENEX system where certain non-printing characters are normally converted to a Circumflex (up-arrow) followed by a printing character.

d. Binary transmission to a terminal.

The implementer of the binary transmission option should consider how (or whether) all characters received over a connection with binary transmission in effect are sent to a local terminal. At issue may be the addition of timing characters normally inserted locally, parity calculations, and any normal code conversion.

TELNET ECHO OPTION  
NIC 15390 (Aug. 1973)

TELNET ECHO OPTION

1. Command name and code.

ECHO 1

2. Command meanings.

IAC WILL ECHO

The sender of this command REQUESTS to begin, or confirms that it will now begin, echoing data characters it receives over the TELNET connection back to the sender of the data characters.

IAC WON'T ECHO

The sender of this command DEMANDS to stop, or refuses to start, echoing the data characters it receives over the TELNET connection back to the sender of the data characters.

IAC DO ECHO

The sender of this command REQUESTS that the receiver of this command begin echoing, or confirms the receiver of this command is expected to echo, data characters it receives over the TELNET connection back to the sender.

IAC DON'T ECHO

The sender of this command DEMANDS the receiver of this command stop, or not start, echoing data characters it receives over the TELNET connection.

3. Default.

WON'T ECHO

DON'T ECHO

i.e., no echoing (which may or may not imply local echoing).

4. Motivation for the option.

The NVT has a printer and a keyboard which are nominally interconnected so that "echoes" need never traverse the network; that is to say, the NVT nominally operates in a mode where characters

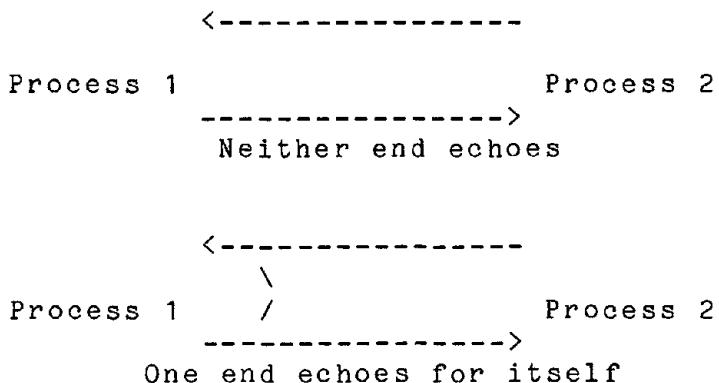
TELNET ECHO OPTION  
NIC 15390 (Aug. 1973)

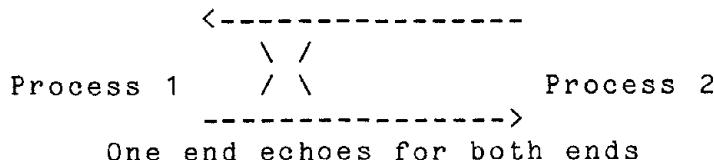
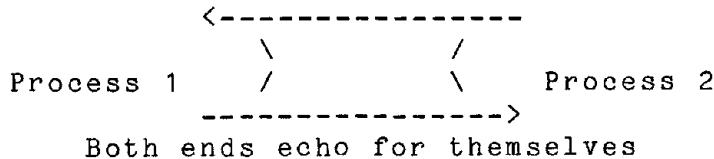
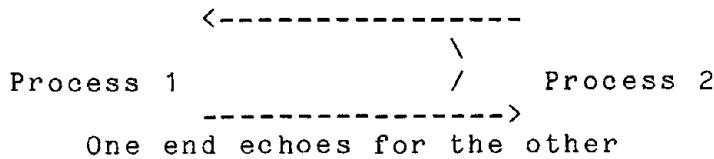
typed on the keyboard are (by some means) locally turned around and printed on the printer. In highly interactive situations it is appropriate for the remote process (command language interpreter, etc.) to which the characters are being sent to control the way they are echoed on the printer. In order to support such interactive situations, it is necessary that there be a TELNET option to allow the parties at the two ends of the TELNET connection to agree that characters typed on an NVT keyboard are to be echoed by the party at the other end of the TELNET connection.

5. Description of the option.

When the echoing option is in effect, the party at the end performing the echoing is expected to transmit (echo) data characters it receives back to the sender of the data characters. The option does not require that the characters echoed be exactly the characters received (for example, a number of systems echo the ASCII ESC character with something other than the ESC character). When the echoing option is not in effect, the receiver of data characters should not echo them back to the sender; this, of course, does not prevent the receiver from responding to data characters received.

The normal TELNET connection uses two coupled, simplex connections, one in each direction; and neither, either, or both of these simplex connections may be operating simultaneously in echo mode. There are five reasonable modes of operation for echoing on a connection pair:





This option provides the capability to decide on whether or not either end will echo for the other. It does not, however, provide any control over whether or not an end echoes for itself; this decision must be left to the sole discretion of the systems at each end (although they may use information regarding the state of "remote" echoing negotiations in making this decision).

It should be noted that if BOTH sites enter the mode of echoing characters transmitted by the other site, then any character transmitted in either direction will be "echoed" back and forth indefinitely. Therefore, care should be taken in each implementation that if one site is echoing, echoing is not permitted to be turned on at the other.

As discussed in the TELNET Protocol Specification, both parties to a full-duplex pair of TELNET connections initially assume each simplex connection is being operated in the default mode which is non-echo. If either party desires himself to echo characters to the other party or for the other party to echo characters to him, that party gives the appropriate command (WILL ECHO or DO ECHO) and waits (and hopes) for acceptance of the option. If the request to operate the connection in echo mode is refused, then the connection continues to operate in non-echo mode. If the request to operate the connection in echo mode is accepted, the connection is operated in echo mode.

TELNET ECHO OPTION  
NIC 15390 (Aug. 1973)

After a connection has been changed to echo mode, either party may demand that it revert to non-echo mode by giving the appropriate DON'T ECHO or WON'T ECHO command (which the other party must confirm thereby allowing the connection to operate in non-echo mode). Just as each of the pair of simplex TELNET connections may be put in remote echoing mode independently, each of the pair of simplex TELNET connections must be removed from remote echoing mode separately.

Implementations of the echo option, as implementations of all other TELNET options, must follow the loop preventing rules given in the General Considerations section of the TELNET Protocol Specification. Also, so switches between echo and non-echo mode can be made with minimal confusion (momentary double echoing, etc.), switches in mode of operation should be made at times precisely coordinated with the reception and transmission of echo requests and demands. For instance, if one party responds to a DO ECHO with a WILL ECHO, all data characters received after the DO ECHO should be echoed and the WILL ECHO should immediately precede the first of the echoed characters.

The echoing option alone will normally not be sufficient to effect what is commonly understood to be remote computer echoing of characters typed on a terminal keyboard--the SUPPRESS-GO AHEAD option will normally have to be invoked in conjunction with the ECHO option to effect character-at-a-time remote echoing.

## 6. A Sample Implementation of the Option.

It may be useful to present a sample implementation--the TIP development group suggests the following:

For each user terminal the TIP would keep three state bits: whether the terminal echoes for itself (no ECHO always) or not (ECHO mode possible), whether the (human) user prefers to operate in ECHO mode or in non-ECHO mode, and whether the connection from this terminal to the server is in ECHO or non-ECHO mode. We call these three bits P(physical), D(esired), and A(ctual).

When a terminal dials up the TIP the P-bit is set appropriately, the D-bit is set equal to it, and the A-bit is set to non-ECHO. The P- and A-bits may be manually reset by direct commands if the user so desires; for instance, a user in Hawaii on a 'full-duplex' terminal might know that whatever the preference of a mainland server, because of satellite delay his terminal had better not operate in ECHO mode - he would direct the TIP to change his D-bit from ECHO to non-ECHO.

TELNET ECHO OPTION  
NIC 15390 (Aug. 1973)

When a connection is opened from the TIP terminal to a server, the TIP would send the server a DO ECHO command if the MIN (with non-ECHO less than ECHO) of the P- and D-bits is different from the A-bit. If a WON'T ECHO or WILL ECHO arrives from the server, the TIP will set the A-bit to the MIN of the received request, the P-bit, and the D-bit. If this changes the state of the A-bit, the TIP will send off the appropriate acknowledgment; if it does not, then the TIP will send off the appropriate refusal if not changing meant that it had to deny the request (i.e., the MIN of the P-and D-bits was less than the received A-request). If while a connection is open, the TIP terminal user changes either the P- or D-bit, the TIP will repeat the above tests and send off a DO ECHO or DON'T ECHO, if necessary. When the connection is closed, the TIP would reset the A-bit to indicate no remote echoing.

While the TIP's implementation would not involve DO ECHO or DON'T ECHO commands being sent to the server except when the connection is opened or the user explicitly changes his echoing mode, bigger Hosts might invoke such mode switches quite frequently. For instance, if JOSS, a line-at-a-time system, were running, the server might attempt to put the user in local echo mode by sending the WON'T ECHO command to the user; but while DDT was running, the server might attempt to invoke remote echoing for the user by sending the WILL ECHO command to the user. Furthermore, while the TIP will never send a WILL ECHO command and will only send a WON'T ECHO to refuse a server sent DO ECHO command, a server Host will often send the WILL and WON'T ECHO commands.



TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

TELNET Reconnection Option

1. Command name and code

RCP                    2                 (prepare to reconnect)

2. Command meanings.

IAC DO RCP

The sender of this command requests the receiver of the command to be prepared to break the TELNET connection with the sender of the command and to re-establish the TELNET connection with some other party (to be specified later).

IAC WILL RCP

The receiver of this command agrees to break its TELNET connection to the sender of the DO RCP command and to re-establish the connection with the party to be specified by the sender of the DO RCP command.

IAC WON'T RCP

The receiver of this command refuses to take part in a reconnection.

IAC DON'T RCP

The sender of this command demands the cancellation of its previous DO RCP command.

IAC SB RCP RCS <host> <socket>

The sender of this command instructs the receiver of the command to transfer this TELNET connection to the place specified by <host> <socket>. The code for RCS is 0.

IAC SB RCP RCW <host> <socket>

The sender of this command instructs the receiver of the command to break the TELNET connection and to await a new TELNET connection from the place specified by <host> <socket>. The code for RCW is 1.

Preceding page blank

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

3. Default.

WON'T RCP

i.e., no reconnection is allowed.

4. Motivation for the option.

There are situations in which it is desirable to move one or both ends of a communication path from one Host to another.

A. Consider the case of an executive program which TIP users could use to get network status information, send messages, link to other users, etc. Due to the TIP's limited resources the executive program would probably not run on the TIP itself but rather would run on one or more larger Hosts who would be willing to share some of their resources with the TIP (see Figure 1).

The TIP user could access the executive by typing a command such as "@EXEC"; the TIP should then ICP to Host1's executive port. After obtaining the latest network news and perhaps sending a few messages, the user would be ready to log into Host2 (in general not the same as Host1) and do some work. At that point he would like to tell the executive program that he is ready to use Host2 and have the executive hand him off to Host2. To do this the executive program would first interact with Host2, telling it to expect a call from the TIP, and then would instruct the TIP to reconnect to Host2. When the user logs off Host2 he could be passed back to the executive at Host1 preparatory to doing more elsewhere. The reconnection activity would be invisible to the TIP user.

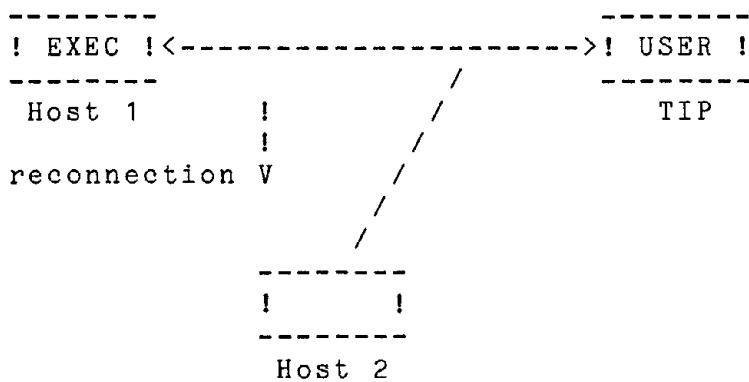


FIGURE 1

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

B. Imagine a scenario in which a user could use the same name and password (and perhaps account) to log into any server on the network. For reasons of security and economy it would be undesirable to have every name and password stored at every site. A user wanting to use a Host that doesn't have his name or password locally would connect to it and attempt to log in as usual (see Figure 2). The Host, discovering that it doesn't know the user, would hand him off to a network authentication service which can determine whether the user is who he claims to be. If the user passes the authentication test he can be handed back to the Host which can then provide him service.

If the user doesn't trust the Host and is afraid that it might read his password rather than pass him off to the Authenticator he could connect directly to the authentication service. After authentication, the Authenticator can pass him off to the Host.

The idea is that the shuffling of the user back and forth between Host and Authenticator should be invisible to the user.

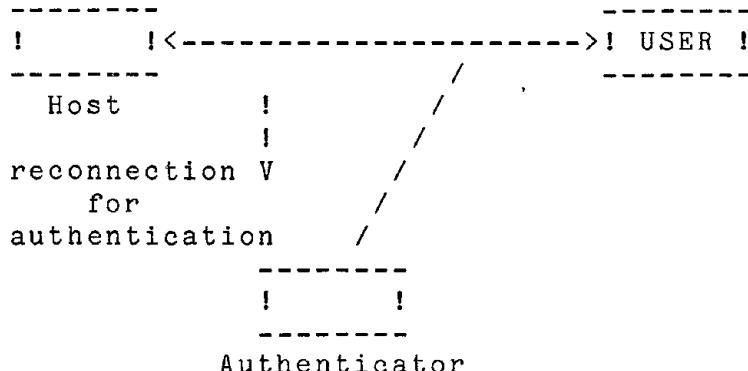


FIGURE 2a

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

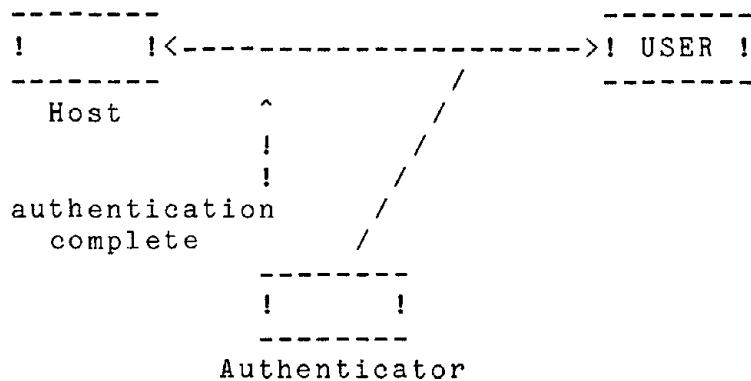


FIGURE 2b

C. The McROSS air traffic simulation system (see 1972 SJCC paper by Thomas) already supports reconnection. It permits an on-going simulation to reconfigure itself by allowing parts to move from computer to computer. For example, in a simulation of air traffic in the Northeast, the program fragment simulating the New York Enroute air space could move from Host2 to Host5 (see figure 3). As part of the reconfiguration process the New York Terminal area simulator and Boston Enroute area simulators break their connections with the New York Enroute simulator at Host2 and reconnect to it at Host5.

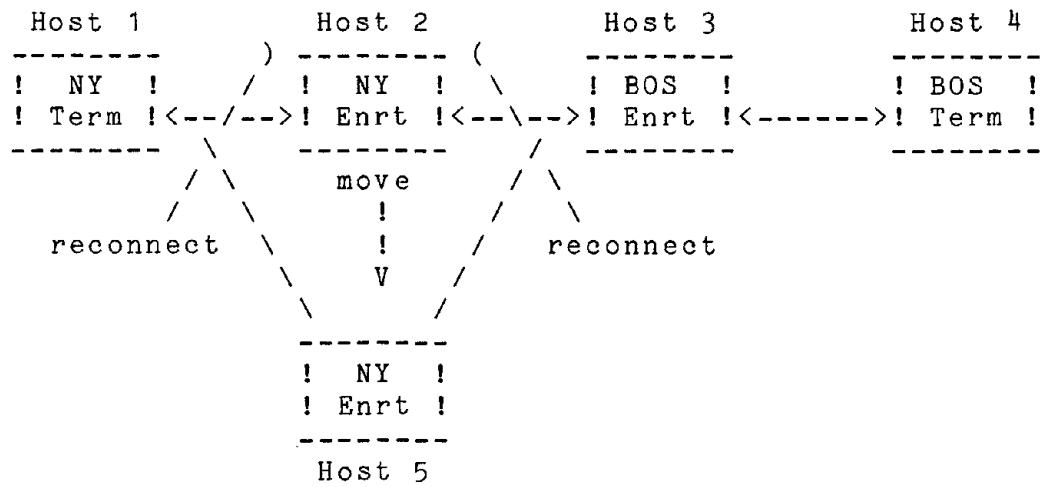


FIGURE 3

5. An abstract description of a reconnection mechanism.

The reconnection mechanism includes four (abstract) commands:

```
Reconnect Request: RRQ <path>
Reconnect OK: ROK <path>
Reconnect No: RNO <path>
Reconnect Do: RDO <path> <destination>
```

where <path> is a communication path to be redirected to <destination>.

Assume that H1 wants to move its end of communication path A-C from itself to port D at H3 (Figure 4).

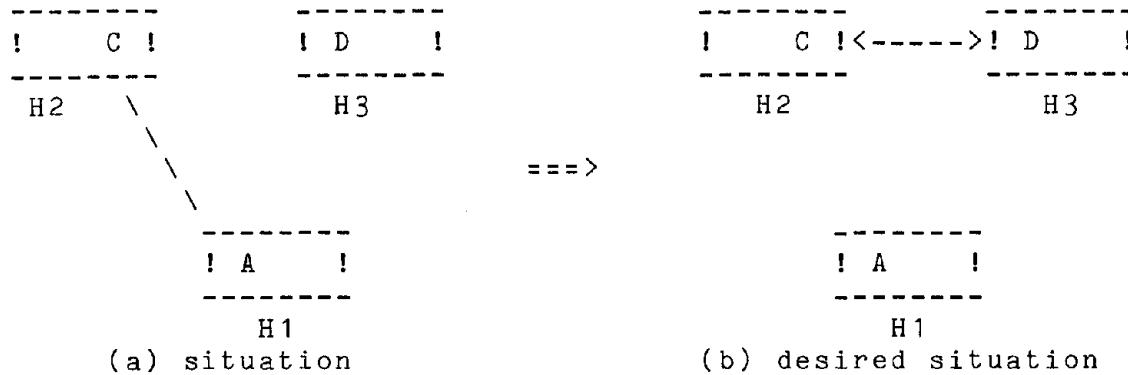


FIGURE 4

The reconnection proceeds by steps:

a. H1 arranges for the reconnection by sending RRQ to H2:

H1->H2: RRQ (path A-C)

b. H2 agrees to reconnect and acknowledges with ROK:

H2->H1: ROK (path C-A)

c. H1 notes that H2 has agreed to reconnect and instructs H2 to perform the reconnection;

H1->H2: RDO (path A-C) (Host H3, PortD)

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

d. H1 breaks paths A-C.

H2 breaks path C-A and initiates path C-D.

In order for the reconnection to succeed H1 must, of course, have arranged for H3's cooperation. One way H1 could do this would be to establish the path B-D and then proceed through the reconnection protocol exchange with H3 concurrently with its exchange with H2 (See Figure 5):

H1->H3: RRQ (path B-D)  
H3->H1: ROK (path D-B)  
H1->H3: RDO (path B-D) (Host H2, Port C)

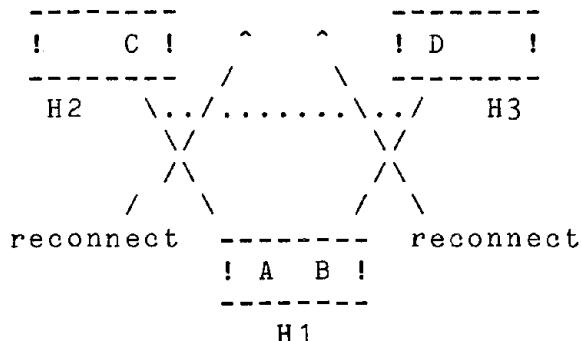


FIGURE 5

Either of the parties may use the RNO command to refuse or abort reconnection. H2 could respond to H1's RRQ with RNO; H1 can abort the reconnection by responding to ROK with RNO rather than RDO.

It is easy to insure that messages in transit are not lost during the reconnection. Receipt of the ROK message by H1 is taken to mean that no further messages are coming from H2; similarly receipt of RDO from H1 by H2 is taken to mean that no further messages are coming from H1.

To complete the specification of the reconnection mechanism consider the situation in which two "adjacent" entities initiate reconnections:

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

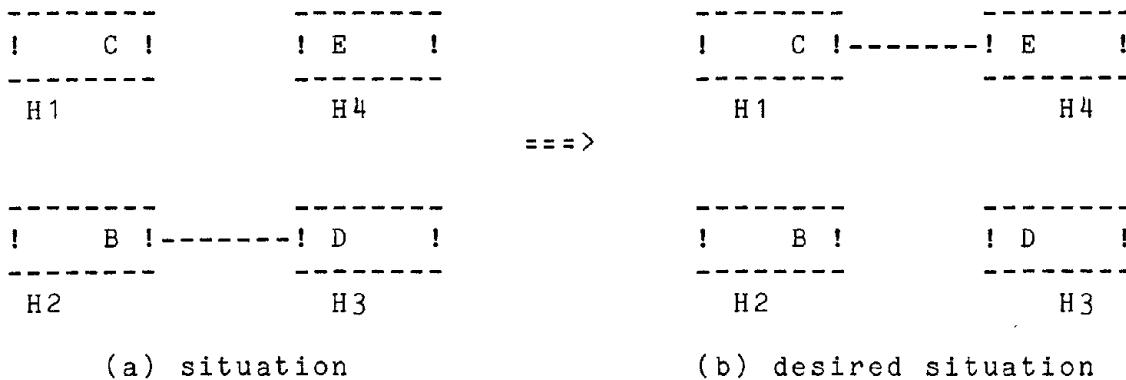


FIGURE 6

H2 and H3 "simultaneously" try to arrange for reconnection:

H2->H3: RRQ (path B-D)

H3->H2: RRQ (path D-B)

Thus, H2 sees an RRQ from H3 rather than an ROK or RNO in response to its RRQ to H3. This "race" situation can be resolved by having the reconnections proceed in series rather than in parallel: first one entity (say H2) performs its reconnect and then the other (H3) performs its reconnect. There are several means that could be used to decide which gets to go first. Perhaps the simplest is to base the decision on sockets and site addresses: the entity for which the 40 bit number formed by concatenating the 32 bit socket number with the 8 bit site address is largest gets to go first. Using this mechanism the rule is the following:

If H2 receives an RRQ from H3 in response to an RRQ of its own:

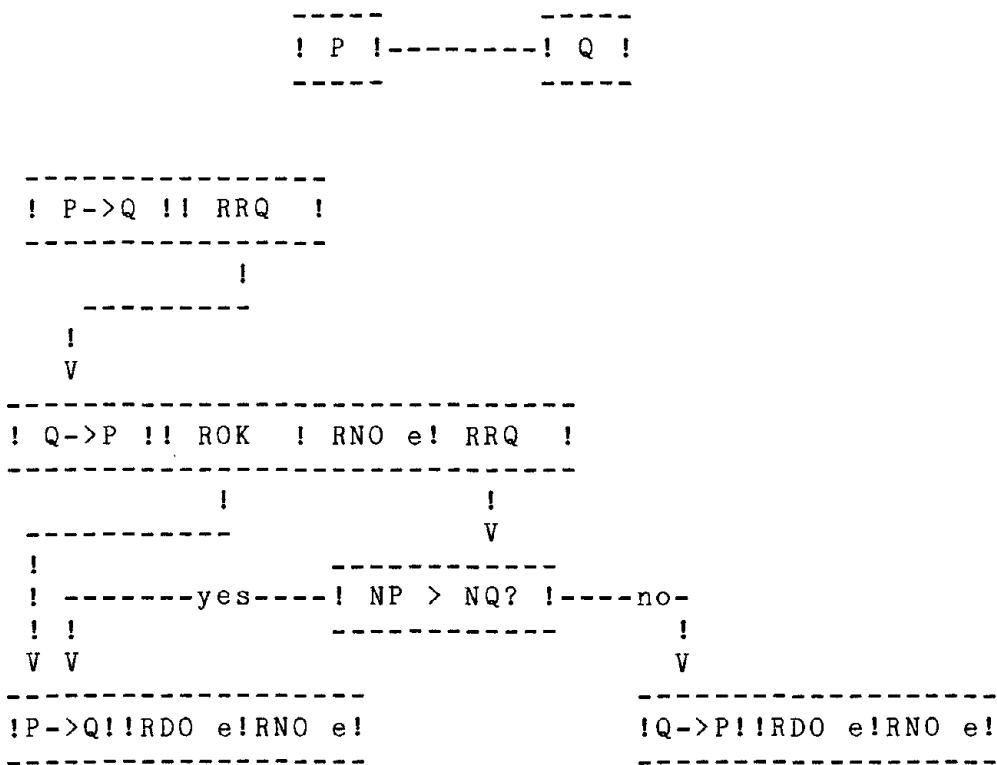
(let NH2, NH3 = the 40 bit numbers corresponding to H2 and H3)

- a. if  $NH2 > NH3$  then both H2 and H3 interpret H3's RRQ as an ROK in response to H2's RRQ.
- b. if  $NH2 < NH3$  then both interpret H3's RRQ as an RNO in response to H2's RRQ. This would be the only case in which it makes sense to "ignore" the refusal and try again - of course, waiting until completion of the first reconnect before doing so.

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

Once an ordering has been determined the reconnection proceeds as though there was no conflict.

The following diagram describes the legal protocol command/response exchange sequences for a reconnection initiated by P:



NP and NQ are the 40 bit numbers for P and Q; e indicates end of sequence.

## 6. A description of the option.

The reconnection mechanism described abstractly in the previous section can be effected as a TELNET option by use of the command RCP. Using this command and the TELNET DO, DON'T, WILL, WON'T, and SB prefixes, the four commands used in the previous abstract description become

RRQ => DO RCP

ROK => WILL RCP

RNO => WON'T RCP ;for responses to DO RCP

TELNET Reconnection Option  
NIC 15391 (Aug. 1973)

DON'T RCP ;for responses to WILL RCP  
;i.e. used to cancel an RCP.

RDO <host> <socket> => SB RCP RCS <host> <socket>

A fifth command is also introduced

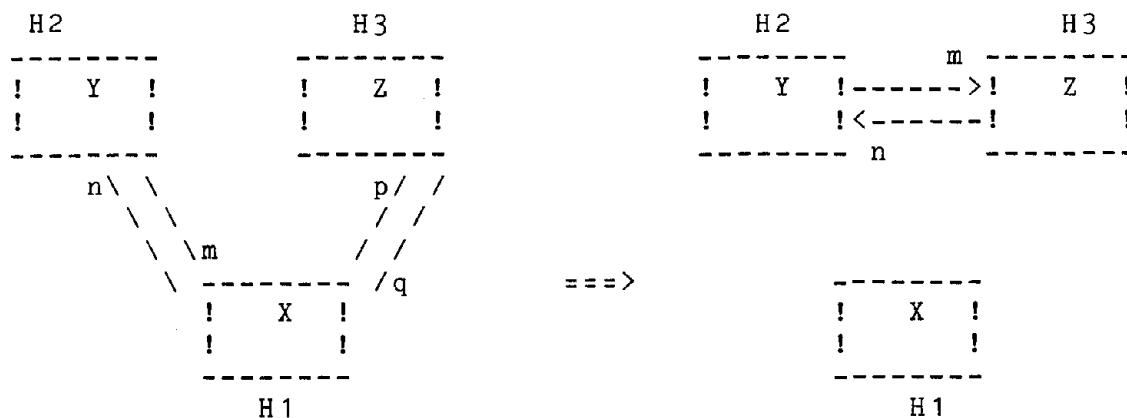
RWT <host> <socket> => SB RCP RCW <host> <socket>

The first three commands require no parameters since they refer to the connections they are received on. For RDO and RWT, <host> is an 8 bit (= 1 TELNET character) Host address and <socket> is a 32 bit (= 4 TELNET characters) number that specifies a TELNET receive socket at the specified Host (the associated transmit socket is always one higher than the receive socket).

A pending reconnection can be activated with either RDO or RWT. The response to either is to first break the TELNET connection with the sender and then reopen the TELNET connection to the Host and sockets specified. For RDO, the connection is to be reopened by sending two RFC's; for RWT, by waiting for two RFC's.

The RWT command is introduced to avoid requiring Hosts to queue RFC's.

As an example, the reconnection



could be accomplished as follows:

X->Y: RRQ (=IAC DO RCP)  
X->Z: RRQ (=IAC DO RCP)

```
Y->X: ROK      (=IAC WILL RCP)
Z->X: ROK      (=IAC WILL RCP)
X->Y: RWT H3 P  (=IAC SB RCP RCW H3 P)
X closes connections to Y
Y closes connections to X
Y waits for STR and RTS from H3
X->Z: RDO H2 N  (=IAC SB RCP RCS H2 N)
X closes connections to Z
Z closes connections to X
Z sends STR and RTS to H2 which Y answers with matching RTS and
STR to compete reconnection
```

The RCS and RCW sub-commands should never be sent until a DO RCP has been acknowledged by a WILL RCP. Thus a Host not choosing to implement the reconnection option does not have to know what RCP means--all the Host need do in response to DO RCP is to transmit WON'T RCP. The WILL RCP and WON'T RCP commands should never be volunteered. If an unsolicited WILL RCP is ever received, a DON'T RCP should be fired back, which should be answered by a WON'T RCP command. If an unsolicited WON'T RCP command is received, it should be treated as a No-operation.

#### 7. A word about security.

It should be clear that the decision to accept or reject a particular reconnection request is the responsibility of the entity (person at the terminal or process) using the connection. In many cases the entity may chose to delegate that responsibility to its TELNET (e.g., Example A, Section 4). However, the interface a Host provides to the reconnection mechanism would best include means for local entities to exercise control over response to remotely intitiated reconnection requests. For example, a user-TELNET might support several modes of operation with respect to remotely initiated reconnectons:

1. transparent: all requested reconnections are to be performed in a way that is invisible to the user;
2. visible: all requested reconnections are to be performed and the user is to be informed whenever a reconnection occurs;
3. confirmation: the user is to be informed of each reconnection request which he may accept or reject;
4. rejection: all requested reconnects are to be rejected.

TELNET SUPPRESS GO AHEAD OPTION  
NIC 15392 (Aug. 1973)

TELNET SUPPRESS GO AHEAD OPTION

1. Command name and code.

SUPPRESS-GO-AHEAD 3

2. Command meanings.

IAC WILL SUPPRESS-GO-AHEAD

The sender of this command requests permission to begin suppressing transmission of the TELNET GO AHEAD (GA) character when transmitting data characters, or the sender of this command confirms it will now begin suppressing transmission of GAs with transmitted data characters.

IAC WON'T SUPPRESS-GO-AHEAD

The sender of this command demands to begin transmitting, or to continue transmitting, the GA character when transmitting data characters.

IAC DO SUPPRESS-GO-AHEAD

The sender of this command requests that the sender of data start suppressing GA when transmitting data, or the sender of this command confirms that the sender of data is expected to suppress transmission of GAs.

IAC DON'T SUPPRESS-GO-AHEAD

The sender of this command demands that the receiver of the command start or continue transmitting GAs when transmitting data.

3. Default.

WON'T SUPPRESS-GO-AHEAD

DON'T SUPPRESS-GO-AHEAD

i.e., transmit GAs.

4. Motivation for the option.

While the NVT nominally follows a half duplex protocol complete with a GO AHEAD signal, there is no reason why a full duplex connection between a full duplex terminal and a Host optimized to handle full

TELNET SUPPRESS GO AHEAD OPTION  
NIC 15392 (Aug. 1973)

duplex terminals should be burdened with the GO AHEAD signal. Therefore, it is desirable to have a TELNET option with which parties involved can agree that one or the other or both should suppress transmission of GO AHEADS.

5. Description of the option.

When the SUPPRESS-GO-AHEAD option is in effect on the connection between a sender of data and the receiver of the data, the sender need not transmit GAs.

It seems probable that the parties to the TELNET connection will suppress GO AHEAD in both directions of the TELNET connection if GO AHEAD is suppressed at all; but, nonetheless, it must be suppressed in both directions independently.

With the SUPPRESS-GO-AHEAD option in effect, the IAC GA command should be treated as a NOP if received, although IAC GA should not normally be sent in this mode.

6. Implementation considerations.

As the SUPPRESS-GO-AHEAD option is sort of the opposite of a line at a time mode, the sender of data which is suppressing GO AHEADs should attempt to actually transmit characters as soon as possible (i.e., with minimal buffering) consistent with any other agreements which are in effect (e.g., approximate transmit message size agreements).

In many TELNET implementations it will be desirable to couple the SUPPRESS-GO-AHEAD option to the echo option so that when the echo option is in effect, the SUPPRESS-GO-AHEAD option is in effect simultaneously: both of these options will normally have to be in effect simultaneously to effect what is commonly understood to be character at a time echoing by the remote computer.

TELNET Approximate Message Size Negotiation Option  
NIC 15393 (Aug. 1973)

TELNET Approximate Message Size Negotiation Option

1. Command name and code.

NAMS                  4

(Negotiate Approximate Message Size)

2. Command meanings.

IAC WILL NAMS

The sender of this command requests, or agrees, to negotiate the approximate size for messages of data characters it sends.

IAC WON'T NAMS

The sender of this command refuses to negotiate the approximate size for messages of data characters it sends.

IAC DO NAMS

The sender of this command requests the receiver of this command to negotiate the approximate size for messages of data characters transmitted by the command receiver.

IAC DON'T NAMS

The sender of this command refuses to negotiate the approximate size for messages of data characters transmitted by the command receiver.

IAC SB NAMS DR <16 bit value>

The sender of this command requests the receiver of this command to set its approximate message size for data the command receiver transmits to the value specified in the 16 bit parameter, a data character count. The code for DR (Data Receiver) is 0.

IAC SB NAMS DS <16 bit value>

The sender of this command requests or agrees to set its approximate message size for data it transmits to the value specified in the 16 bit parameter, a data character count. The code for DS (Data Sender) is 1.

TELNET Approximate Message Size Negotiation Option  
NIC 15393 (Aug. 1973)

3. Default

WON'T NAMS

DON'T NAMS

i.e., no attempt will be made to agree on a message size.

4. Motivation for the option.

The TELNET protocol does not specify how many characters the transmitter of data should attempt to pack into messages it sends. However, 1) some receivers may prefer received messages to generally have some minimum size, for example, to lessen the burden of processing input interrupts; 2) some receivers may prefer received data messages to generally have some maximum size, for example, because the maximum data message size could be used in conjunction with the Host/Host protocol message and bit allocates to more efficiently utilize input buffer space; 3) some transmitters may have maximum sizes for transmitted data messages, information which could be used in conjunction with the Host/Host protocol message and bit allocates to more efficiently utilize the receiver's input buffer space; and 4) some transmitters may desire to transmit some minimum size message, for example, to lessen the burden of processing output interrupts.

Therefore, it is desirable to have some mechanism whereby the parties involved can attempt to agree on the approximate size of messages transmitted over the connection. (It might be even more powerful to be able to negotiate approximate or even exact upper and lower bounds on message size. However, fixed bounds would sometimes be hard to manage and sometimes even in conflict with Host/Host protocol allocates; and specifying both upper and lower bounds, even approximately, seems overly complicated considering the expected payoff.)

5. Description of the option.

With the option which specifies the approximate size of messages transmitted over the connection, the transmitter attempts to send messages of the specified size unless some other constraint (for instance, an end of line) requires the message to be sent sooner, or characters for transmission arrive so fast that the message has to be bigger than the specified size. The option is to be used strictly to improve the STATISTICS (e.g., timing and buffering) of message reception and transmission -- the option does NOT specify any absolutes.

TELNET Approximate Message Size Negotiation Option  
NIC 15393 (Aug. 1973)

With this option not in effect, message size is completely (even statistically) undefined as per the NVT specification.

Once the data transmitter and receiver have agreed to negotiate the approximate message size, they must actually do this negotiation. This is done using the DS and DR SB commands. The transmitter of data messages may give the SB NAMS DS command and the receiver may give the SB NAMS DR command. The rules for negotiation of the actual approximate message size are as follows:

- a) Either party may at any time send a SB command specifying a value less than any previously sent or received and immediately assume that that value has been agreed upon.
- b) If either party receives a SB command, the party should assume the value specified in the received command is in effect if the party has not previously sent a SB command specifying a lower value.
- c) Before any SB command is sent, the approximate message size is undefined.
- d) At any time either party may quit the whole thing by sending a DON'T or WON'T NAMS command which must be acknowledged and the approximate message length becomes undefined.
- e) An approximate message size value may not be less than one.

As the receiver and transmitter may have conflicting requirements for the approximate message size, neither should be cavalier about requesting a specified approximate message size, each "bending over backward" to let the other party (who should be presumed to have a greater need) specify the approximate message size.

Host/Host protocol allocate considerations, of course, always dominate negotiated message size considerations.



David Crocker (UC Irvine)  
RFC 651, NIC 31154 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>STATUS-OPTION-REVISION.RNO

Revised TELNET Status Option

1. Command name and code

STATUS 5

2. Command meanings

As described in the NAOL and NAOP option specifications, this option applies to a simplex connection.

IAC DO STATUS

Sender of DO wishes to be able to send requests for status-of-options information, or confirms that he is willing to send such requests.

IAC WILL STATUS

Sender of WILL wishes or agrees to send status information, spontaneously or in response to future requests.

IAC DON'T STATUS

Sender refuses to carry on any further discussion of the current status of options.

IAC WON'T STATUS

Sender refuses to carry on any further discussion of the current status of options.

IAC SB STATUS SEND IAC SE

Sender requests receiver to transmit his (the receiver's) perception of the current status of Telnet options. The code for SEND is 1. (See below.)

---

Preceding page blank

IAC SB STATUS IS ... IAC SE

Sender is stating his perception of the current status of Telnet options. The code for IS is 0. (See below.)

### 3. Default

DON'T STATUS/WON'T STATUS.

That is, the current status of options will not be discussed.

### 4. Motivation for the option

This option allows a user/process to verify the current status of Telnet options (e.g., echoing) as viewed by the person/process on the other end of the Telnet connection. Simply renegotiating options could lead to the nonterminating request loop problem discussed in (NIC #16237). The changes to the option, described in this paper, allow STATUS to fit into the normal structure of Telnet options, by deferring the actual transfer of status information to the SB command. Additionally, the numbers of bytes that must be sent to describe the state of the options has been considerably reduced.

### 5. Description of the option

WILL/DO are now used only to obtain and grant permission for future discussion. The actual exchange of status information occurs within option subcommands (IAC SB STATUS...).

Once the two hosts have exchanged a WILL and a DO, the sender of the WILL STATUS is free to transmit status information, spontaneously or in response to a request from the sender of the DO. At worst, this may lead to transmitting the information twice. Only the sender of the DO may send requests (IAC SB STATUS SEND IAC SE) and only the sender of the WILL may transmit actual status information (within an IAC SB STATUS IS ... IAC SE command).

IS has the subcommands WILL, DO and SB. They are used EXACTLY as used during the actual negotiation of Telnet options, except that SB is terminated with SE, rather than IAC SE. Transmission of SE, as a regular data byte, is accomplished by doubling the byte (SE SE). Options that are not explicitly described are assumed to be in their default states. A single IAC SB STATUS IS ... IAC SE describes the condition of ALL options.

Revised Telnet Status Option  
RFC 651, NIC 31154 (Oct. 25, 1977)

The following is an example of use of the option:

Host1: IAC DO STATUS

Host2: IAC WILL STATUS

(Host2 is now free to send status information at any time.  
Solicitations from Host1 are NOT necessary. This should not  
produce any dangerous race conditions. At worst, two IS's will  
be sent.

Host1 (perhaps): IAC SB STATUS SEND IAC SE

Host2 (the following stream is broken into multiple lines only for  
readability. No carriage returns are implied.):

IAC SB STATUS IS

WILL ECHO

DO SUPPRESS-GO-AHEAD

WILL STATUS

DO STATUS

WILL RCTE

SB RCTE <11><1><24> SE

DO NAOL

SB NAOL DS <66> SE

IAC SE

Explanation of Host2's perceptions: It is responsible for echoing  
back the data characters it receives over the Telnet connection;  
it will not send Go-Ahead signals; it will both issue and request  
Status information; it will send instruction for controlling the  
other side's terminal printer; it will discuss the line width for  
data it is sending.



TELNET TIMING MARK OPTION  
NIC 16238

TELNET TIMING MARK OPTION

1. Command name and code.

TIMING-MARK 6

2. Command meanings.

IAC DO TIMING-MARK

The sender of this command REQUESTS that the receiver of this command return a WILL TIMING-MARK in the data stream at the "appropriate place" as defined in section 4 below.

IAC WILL TIMING-MARK

The sender of this command ASSURES the receiver of this command that it is inserted in the data stream at the "appropriate place" to insure synchronization with a DO TIMING-MARK transmitted by the receiver of this command.

IAC WON'T TIMING-MARK

The sender of this command REFUSES to insure that this command is inserted in the data stream at the "appropriate place" to insure synchronization.

IAC DON'T TIMING-MARK

The sender of this command notifies the receiver of this command that a WILL TIMING-MARK (previously transmitted by the receiver of this command) has been IGNORED.

3. Default

WON'T TIMING-MARK, DON'T TIMING-MARK

i.e., No explicit attempt is made to synchronize the activities at the two ends of the TELNET connection.

4. Motivation for the option

It is sometimes useful for a user or process at one end of a TELNET connection to be sure that previously transmitted data has been completely processed, printed, discarded, or otherwise disposed of. This option provides a mechanism for doing this. In addition, even

Preceding page blank

TELNET TIMING MARK OPTION  
NIC 16238

if the option request (DO TIMING-MARK) is refused (by WON'T TIMING-MARK) the requester is at least assured that the refuser has received (if not processed) all previous data.

As an example of a particular application, imagine a TELNET connection between a physically full duplex terminal and a "full duplex" server system which permits the user to "type ahead" while the server is processing previous user input. Suppose that both sides have agreed to Suppress GA and that the server has agreed to provide echoes. The server now discovers a command which it cannot parse, perhaps because of a user typing error. It would like to throw away all of the user's "type-ahead" (since failure of the parsing of one command is likely to lead to incorrect results if subsequent commands are executed), send the user an error message, and resume interpretation of commands which the user typed after seeing the error message. If the user were local, the system would be able to discard buffered input; but input may be buffered in the user's Host or elsewhere. Therefore, the server might send a DO TIMING-MARK and hope to receive a WILL TIMING-MARK from the user at the "appropriate place" in the data stream.

The "appropriate place", therefore (in absence of other information) is clearly just before the first character which the user typed after seeing the error message. That is, it should appear that the timing mark was "printed" on the user's terminal and that, in response, the user typed an answering timing mark.

Next, suppose that the user in the example above realized that he had misspelled a command, realized that the server would send a DO TIMING-MARK, and wanted to start "typing ahead" again without waiting for this to occur. He might then instruct his own system to send a WILL TIMING-MARK to the server and then begin "typing ahead" again. (Implementers should remember that the user's own system must remember that it sent the WILL TIMING-MARK so as to discard the DO/DON'T TIMING-MARK when it eventually arrives.) Thus, in this case the "appropriate place" for the insertion of the WILL TIMING-MARK is the place defined by the user.

It should be noted, in both of the examples above, that it is the responsibility of the system which transmits the DO TIMING-MARK to discard any unwanted characters; the WILL TIMING-MARK only provides help in deciding which characters are "unwanted".

## 5. Description of the option

Suppose that Process A of Figure 1 wishes to synchronize with B. The DO TIMING-MARK is sent from A to B. B can refuse by replying WON'T

TIMING-MARK, or agree by permitting the timing mark to flow through his "outgoing" buffer, BUF2. Then, instead of delivering it to the terminal, B will enter the mark into his "incoming" buffer BUF1, to flow through toward A. When the mark has propagated through B's incoming buffer, B returns the WILL TIMING-MARK over the TELNET connection to A.

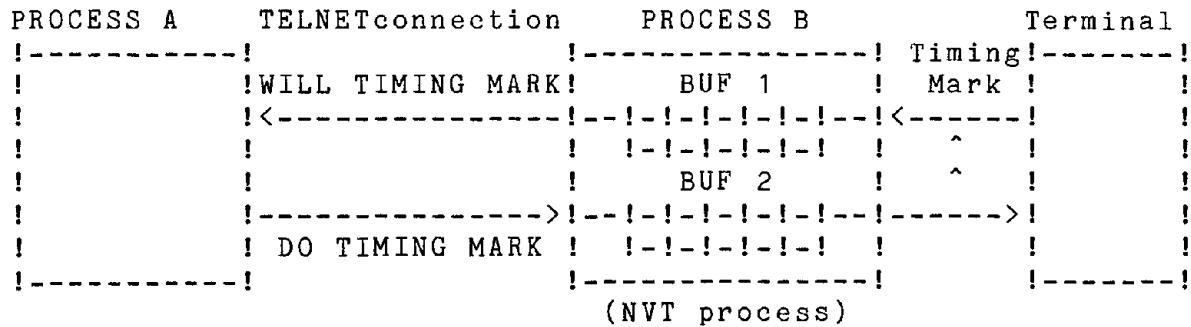


Figure 1

When A receives the WILL TIMING-MARK, he knows that all the information he sent to B before sending the timing markers been delivered, and all the information sent from B to A before turnaround of the timing mark has been delivered.

Three typical applications are:

- A. Measure round-trip delay between a process and a terminal or another process
- B. Resynchronizing an interaction as described in section 4 above. A is a process interpreting commands forwarded from a terminal by B. When A sees an illegal command it:
  - i. Sends <carriage return>, <line feed>, <question mark>
  - ii. Sends DO TIMING-MARK
  - iii. Sends an error message
  - iv. Starts reading input and throwing it away until it receives a WILL TIMING-MARK
  - v. Resumes interpretation of input.

TELNET TIMING MARK OPTION  
NIC 16238

This achieves the effect of flushing all "type ahead" after the erroneous command, up to the point when the user actually saw the question mark.

C. The dual of B above. The terminal user wants to throw away unwanted output from A.

- i. B sends DO TIMING-MARK, followed by some new command.
- ii. B starts reading output from A and throwing it away until it receives WILL TIMING-MARK.
- iii. B resumes forwarding A's output to the terminal.

This achieves the effect of flushing all output from A, up to the point where A saw the timing mark, but not output generated in response to the following command.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

Jon Postel (SRI-ARC)  
Dave Crocker (UC Irvine)  
RFC 726, NIC 39237 (Mar. 8, 1977)  
Obsoletes NIC 18492 and NIC 19859

Remote Controlled Transmission and Echoing Telnet Option

1. Command name and code:

RCTE 7

2. Command meanings:

IAC WILL RCTE

The sender of this command REQUESTS or AGREES to use the RCTE option, and will send instructions for controlling the other side's terminal printer.

IAC WON'T RCTE

The sender of this option REFUSES to send instructions for controlling the other side's terminal printer.

IAC DO RCTE

The sender REQUEST or AGREES to have the other side (sender of WILL RCTE) issue commands which will control his (sender of the DO) output to the terminal printer.

IAC DON'T RCTE

The sender of this command REFUSES to allow the other side to control his (sender of DON'T) terminal printer.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

IAC SB RCTE <cmd> [BC1 BC2] [TC1 TC2] IAC SE

where:

<cmd> is one 8-bit byte having the following flags (bits are counted from the right):

Bit Meaning

- 0 0 = Ignore all other bits in this byte and repeat the last <cmd> that was sent. Equals a 'continue what you have been doing'.
- 1 = Perform actions as indicated by other bits in this byte.
- 1 0 = Print (echo) break character
- 1 = Skip (don't echo) break character
- 2 0 = Print (echo) text up to break character
- 1 = Skip (don't echo) text up to break character
- 3 0 = Continue using same classes of break characters.
- 1 = The two 8-bit bytes following this byte contain flags for the new break classes.
- 4 0 = Continue using same classes of transmit characters.
- 1 = Reset transmit classes according to the two bytes following 1) the break classes bytes, if the break classes are also being reset, or 2) this byte, if the break classes are NOT also being reset.

Value (decimal) of the <cmd> byte and its meaning:

0 = Continue what you have been doing

Even numbers greater than zero (i.e. numbers with the right most bit off) are in error and should be interpreted as equal to zero. When the <cmd> is an even number greater than zero, classes bytes TC1 and TC2 and/or BC1 and BC2 must not be sent.

1 = Print (echo) up to AND INCLUDING break character

3 = Print up to break character and SKIP (don't echo) break character

5 = Skip text (don't echo) up to break character, but PRINT break character

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

7 = Skip up to and including break character

Add one of the previous non-zero values to one of the following values, to get the total decimal value for the byte (Note that classes may not be reset without also resetting the printing action; so an odd number is guaranteed):

8 = Set break classes (using the next two bytes [BC1 BC2])

16 = Set transmission classes (using the next two bytes [TC1 TC2])

24 = Set break classes (using the next two bytes [BC1 BC2]) and the transmission classes (using the two bytes after that [TC1 TC2]).

Sub-commands (IAC SB RCTE...) are only sent by the controlling host and, in addition to other functions, functionally replace the Go-Ahead (IAC GA) Telnet feature. RCTE also functionally replaces the Echo (IAC ECHO) Telnet option. That is the Suppress Go-Ahead option should be in force and the Echo option should not be in force while the RCTE option is in use. The echo mode on terminating use of the RCTE option should be the default state, that is DON'T ECHO, WON'T ECHO.

Classes for break and transmission (the right-most bit of the second byte (TC2 or BC2) represents class 1; the left-most bit of the first byte (TC1 or BC1) represents the currently undefined class 16:

1: Upper-Case Letter (A-Z)

2: Lower-case Letters (a-z)

3: Numbers (0-9)

4: Format Effectors (<BS> <CR> <LF> <FF> <HT> <VT>)

The sequence <cr><lf> counts as one character when processed as the Telnet end of line, and is a single break character when class 4 is set. The sequence <cr><nul> counts as one character and is a break character if and only if <cr> is a break character (i.e. class 4 is set).

5: Non-format Effector Control Characters including <DEL> and <ESC>

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

```
6: . , ; : ? !
7: { [ ( < > ) ] }
8: ' " / \ % @ $ & # + - * = ^ _ | ~
9: <Space>
```

And Telnet commands (IAC . . .) sent by the user are always to have the effect of a break character. That is, every instance of an IAC is to be treated as a break character, except the sequence IAC IAC.

The representation to be displayed when printing is called for is the obvious one for the visible characters (classes 1, 2, 3, 6, 7, and 8). Space (class 9) is represented by a blank space. The format effectors (class 4) by their format effect. The non-format effector controls (class 5) print nothing (no space).

Initially no break classes or transmission classes are in effect.

Please note that if all the bits are set in a Telnet subcommand argument byte such as TC2 or BC2 then that byte must be preceded by an <IAC> flag byte. This is the common convention of doubling the escape character to use its value as data.

Sub-commands (IAC SB RCTE...) are referred to as "break reset commands".

### 3. Default

WON'T RCTE -- DON'T RCTE

Neither host asserts special control over the other host's terminal printer.

### 4. Motivation for the option

RFC's 1, 5, and 51 discuss Network and process efficiency and smoothness.

RFC 357, by John Davidson, introduces the problem of echoing delay that occurs when a remote user accesses a full-duplex host, through a satellite link. In order to save the many thousands of miles of transit time for each echoed character, while still permitting full server responsiveness and clean terminal output, an echo control

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

similar to that used by some time-sharing systems is suggested for the entire Network.

In effect, the option described in this document involves making a using host carefully regulate the local terminal printer according to explicit instructions from the remote (serving) host.

An important additional issue is efficient Network transmission. Implementation of the Davidson Echoing Scheme will eliminate almost all server-to-user echoing.

The option described in this document also requests that using hosts buffer a terminal's input to the serving host until it forms a useful unit (with "useful unit" delimited by break or transmission characters as described below). Therefore, fewer messages are sent on the user-to-server path.

NOTE: This option is only intended for use with full-duplex hosts. The Go-Ahead Telnet feature is completely adequate for half-duplex server hosts. Also, RCTE should be used in place of the ECHO Telnet option, i.e., the Suppress Go-Ahead option should be in force and the Echo option should not be in force while the RCTE option is in use.

## 5. Explicit description of control mechanism

### User Terminal Printing Action and Control Procedure

Negotiate the use of the RCTE option. Once the option is in force the user Telnet follows the following procedure.

1) Read an item from the network.

If the item is data, then print it and go to 1.

If the item is a command, then set the classes and go to 2.

2) If the terminal input buffer is empty, then go to 3, else go to 4.

3) Wait for an item to appear either from the terminal or from the network.

If an item appears from the terminal, then go to 4.

If a data item appears from the network, then print it and go to 3.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

If a command appears from the network, then an error has occurred.

- 4) Read an item from the terminal input buffer.

If the item is not a break, then print/skip it and go to 2.

If the item is a break, then print/skip it and go to 1.

NOTE: Output from the server host may occur at any time, such "spontaneous output" is printed in step 3.

#### Explanation

Both Hosts agree to use the RCTE option. After that, the using host (IAC DO RCTE) merely acts upon the controlling (serving) host's commands and does not issue any RCTE commands unless and until it (using host) decides to stop allowing use of the option (by sending IAC DON'T RCTE).

- 1) The using host is synchronized with the server by initially and when ever it returns to step 1 suspending terminal echo printing until it receives a command from the server.

The server may send either output to the terminal printer or a command, and usually sends a both.

The server may send output to the terminal printer either in response to user input or spontaneously. In the former case, the output is processed in step 1. In the latter case, the output is processed in step 3.

Server sends an RCTE command. The command may redefine break and transmission classes, action to be performed on break characters, and action to be performed on text. Each of these independent functions is controlled by separate bits in the <cmd> byte.

A transmission character is one which RECOMMENDS that the using host transmit all text accumulated up to and including its occurrence. (For network efficiency, using hosts are DISCOURAGED (but not prohibited) from sending before the occurrence of a transmission character, as defined at the moment the character is typed).

If the transmission classes bit (bit 4) is on, the two bytes following the two break classes bytes (or immediately

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

following the <cmd> byte, if the break classes bit is not on) will indicate what classes are to be enabled.

If the bit is OFF, the transmission classes remain unchanged. When the RCTE option is first initiated, NO CLASSES are in effect. That is, no character will be considered a transmission character. (As if both TC1 and TC2 are zero.)

A break character REQUIRES that the using host transmit all text accumulated up to and including its occurrence and also causes the using host to stop its print/discard action upon the user's input text, until directed to do otherwise by another IAC SB RCTE <cmd> IAC SE command from the serving host. Break characters therefore define printing units. "Break character" as used in this document does NOT mean Telnet Break character.

If the break classes bit (bit 3) is on, the two bytes following <cmd> will indicate what classes are to be enabled. There are currently nine (9) classes defined, with room for expansion.

If the bit is OFF, the break classes remain unchanged. When the RCTE option is initiated, NO CLASSES are to be in effect. That is, no transmission will take place in the user to server direction until the first break reset command is received by the user from the server.

The list of character classes, used to define break and transmission classes are listed at the end of this document, in the Tables Section.

Because break characters are special, the print/discard action that should be performed upon them is not always the same as should be performed upon the rest of the input text.

For example, while typing a filename to TENEX, I want the text of the filename to be printed (echoed); but I do not want the <escape> (if I use the name completion feature) to be printed.

If bit 1 is ON the break character is NOT to be printed.

A separate bit (bit 2) signals whether or not the text itself should be printed (echoed) to the terminal. If bit 2 = 0, then the text IS to be printed.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

Yet another bit (bit 0 - right-most bit) signals whether or not any of the other bits of the command should be checked. If this bit is OFF, then the command should be interpreted to mean "continue whatever echoing strategy you have been following, using the same break and transmission classes."

- 2) The user Telnet now checks the terminal input buffer, if it contains data it is processed in step 4, otherwise the user Telnet waits in step 3 for further developments.
- 3) The user Telnet waits until either the human user enters some data in which case Telnet proceeds to step 4, or an item is received from the network. If the item from the network is data it is spontaneous output and is printed, Telnet then continues to wait. If the item from the network is a command then an error has occurred. In this case the user Telnet may attempt to resynchronize the use of RCTE as indicated below.
- 4) Items from the terminal are processed with printing controlled by the settings of the latest break reset command. When a break character is processed, the cycle of control is complete and action re-commences at step 1.

Input from the terminal is (hopefully) buffered into units ending with a transmission or break character; and echoing of input text is suspended after the occurrence of a break character and until receipt of a break reset command from the serving host. The most recent break command determines the break actions.

In summary, what is required is that for every break character sent in the user to server direction there be a break reset command sent in the server to user direction. The user host initially has no knowledge of which characters are break characters, and so starts in a state that assumes that there are no break characters and also that no echoing is to be provided. The server host is expected to send a break reset command to establish the break classes and the echoing mode before it receives any data from the user.

#### Synchronization and Resynchronization

The serving and using hosts must carefully synchronize break reset commands with the transmission of break characters. Except at the beginning of an interaction, the serving host may only send a break reset command in response to the Using host's having sent a break character as defined at that time. This should establish a one-to-one correspondence between them. (A <cmd> value of zero,

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

in this context, is interpreted as a break classes reset to the same class(es) as before.) The break reset command may be preceded by terminal output.

The re-synchronization of the break characters and the break reset commands is done via the exchange of the Telnet signal Abort Output (AO) in the server to user direction and the SYNCH in the user to server direction.

Suppose the server wants to resynchronize the break characters and the break reset commands.

- a. The server should be sure all output to the terminal has been printed by using, for example, the Timing Mark Option.
- b. The server sends the AO signal.
- c. The user receives the AO signal. The user flushes all user to server data whether it has been echoed or not. The user sends a SYNCH to the server. [The SYNCH consists of the Telnet Data Mark (DM) and the host-to-host interrupt (INS).] The user now enters the initial state at step 1.
- d. The server receives the SYNCH and flushes any data preceding the DM (as always). The server now sends a break reset command. (Actually the break reset command could be sent at any time following the AO.)

Suppose the user wants to resynchronize the break characters and the break reset commands.

- a. The user should discard all user to server data whether it has been echoed or not.
- b. The user sends the AO signal. The user now enters the algorithm at step 1.
- c. The server receives the AO signal. The server discards all data buffered but not yet sent to the user. The server sends a SYNCH to the user. The server sends a break reset command to the user.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

Notes and Comments

Even-numbered commands, greater than zero, are in error, since they will have the low-order bit off. The command should be interpreted as equal to zero, which means that any classes reset bytes ([TC1 TC2] [BC1 BC2]) will be in error. (The IAC SE, at the end of the command, eliminates any parsing problems due to this error.)

Serving hosts will generally instruct using hosts not to echo break characters, even though it might be alright to echo most break characters. For example, <cr> is usually a safe character to echo but <esc> is not. TENEX Exec is willing to accept either, during filename specification. Therefore, the using host must be instructed not to echo any break characters.

This is generally a tolerable problem, since the serving host has to send an RCTE command at this point, anyhow. Adding an echo for the break character to the message will not cause any extra network traffic.

The RCTE Option entails a rather large overhead. In a true character-at-a-time situation, this overhead is not justified, but on the average, it should result in significant savings, both in network traffic and host wake-ups.

Buffering Problems and Transmission vs. Printing Constraints:

There are NO mandatory transmission constraints. The using host is allowed to send a character a time, though this would be a waste of RCTE. The transmission classes commands are GUIDELINES, so deviating from them, as when the user's buffer gets full, is allowed.

Additionally, the using host may send a break class character, without knowing that it is one (as with type-ahead).

If the user implementation is clever it may send the user entered data to the server before it is actually needed. This type ahead data may contain break characters.

Assume that only space is a break character (that is the last break reset command specified print up to and including the break characters and set the break classes to class 9). Suppose the user had typed "abc<space>def<esc>ghi<cr>". The user side RCTE could send it all to the server, but it could print only "abc<space>", and would have to buffer

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

"def<esc>ghi<cr>" at least until a break reset command was received from the server. That break reset command could change the break classes, requiring rescanning of the buffered string.

For example, suppose the break reset command set the break characters to class 5 and the action to print up to, but not including, the break character. The user RCTE could then print "def" and discard the <esc>, but would have to continue to buffer the "ghi<cr>".

The problem with buffering occurs when printing on the user's terminal must be suspended, after the user has typed a currently valid break character and until a break reset command is received from the serving host. During this time, the user may be typing merrily along. The text being typed may be SENT, but may not yet be PRINTED.

The more common problem of filling the transmission buffer, while awaiting a host to host allocate from the serving host, may also occur, but this problem is well known to implementors and in no way special to RCTE.

In any case, when the buffer does fill and further text typed by the user will be lost, the user should be notified (perhaps by ringing the terminal bell).

Text should be buffered by the using host until the user types a character which belongs to the transmission class in force at the moment the character is typed.

Transmission class reset commands may be sent by the serving host at any time. If they are frequently sent separate from break class reset commands, it will probably be better to exit from RCTE and enter regular character at a time transmission.

It is not immediately clear what the using host should do with currently buffered text, when a transmission classes reset command is received. The buffering is according to the previous transmission classes scheme.

The using host clearly should not simply wait until a transmission character (according to the new scheme) is typed.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

Either the buffered text should be rescanned, under the new scheme; or the buffered text should simply be sent as a group. This is the simpler approach, and probably quite adequate.

It is possible to define NO BREAK CHARACTERS except Telnet commands (IAC ...). This seems undesirable and should not be done.

If this situation were to occur the using host should send a Telnet command to allow the server to know when he may reset the break classes, but the mechanism is awkward and this case should be avoided.

## 6. Sample Interaction

"S:" is sent from serving (WILL RCTE) host to using host.

"U:" is sent from using (DO RCTE) host to serving host.

"T:" is entered by the terminal user.

"P:" is printed on the terminal.

Text surrounded by square brackets ([]) is commentary.

Text surrounded by angle brackets (<>) is to be taken as a single unit, e.g., carriage return is <cr>, and the decimal value 27 is represented <27>.

The following interaction shows a logon to a Tenex, initiation of the DED editor, insertion of some text and the return to the Exec level.

An attempt has been made to give some flavor of the asynchrony of network I/O and the user's terminal input. Many other possible combinations, using the same set of actions listed below, could be devised. The actual order of events will depend upon network and hosts' load and the user's typing speed.

We assume that the user's Telnet is also in an "insert linefeed" mode. That is, whenever the user types carriage return <cr> the user Telnet sends both carriage return and linefeed <cr><lf> (the Telnet end of line signal). When space character occurs at the end of a line in the example description it is shown explicitly by <sp> to avoid confusion. Other uses of the space character are not so marked to avoid destroying the readability of the example.

A Telnet connection has already been opened, but the TENEX prompt has not yet been issued. The hosts first discuss using the RCTE option:

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

S: <IAC><WILL><RCTE>

U: <IAC><DO><RCTE>

S: TENEX 1.31.18, TENEX EXEC 1.50.2<cr><lf>@  
<IAC><SB><RCTE><11><1><24><IAC><SE>

[Print the herald and echo input text up to a break character,  
but do not echo the break character. Classes 4 (Format  
Effectors), 5 (Non-format Effector Controls and <DEL>), and 9  
(<sp>) act as break characters.]

P: TENEX 1.31.18, TENEX EXEC 1.50.2<cr><lf>@

T: LOGIN ARPA<cr>

P: LOGIN

U: LOGIN<sp>

U: ARPA<cr><lf>

S: <sp><IAC><SB><RCTE><0><IAC>SE>

P: <sp>ARPA

S: <cr><lf>(PASSWORD): <IAC><SB><RCTE><7><IAC><SE>

P: <cr><lf>(PASSWORD):<sp>

T: WASHINGTON 1000<cr>

[The password "WASHINGTON" is not echoed. Printing of  
"1000<cr>" is withheld]

U: WASHINGTON<sp>

U: 1000<cr><lf>

S: <sp><IAC><SB><RCTE><3><IAC><SE>

S: <cr><lf>JOB 17 ON TTY41 7-JUN-73 14:13<cr><lf>@  
<IAC><SB><RCTE><0><IAC><SE>

P: <sp>1000

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

[Printing is slow at this point; so the account number is not printed as soon as the server's command for it is received.]

P: <cr><lf>JOB 17 ON TTY41 7-JUN-73 14:13<cr><lf>@

T: DED<esc><cr>

P: DED

U: DED<esc>

S: .SAV;1<IAC><SB><RCTE><0><IAC><SE>

P: .SAV;1

U: <cr><lf>

S: <cr><lf><lf>DED 3/14/73 DRO,KRK<cr><lf>:  
<IAC><SB><RCTE><15><1><IAC><255><IAC><SE>

[The program is started and the DED prompt ":" is sent. At the command level, DED responds to every character. The server sets the break classes to all classes.]

P: <cr><lf><lf>DED 3/14/73 DRO,KRK<cr><lf>:

T: IThis is a test line.<cr>This is another test line.<^Z>Q

["I" means Insert Text. The text follows, terminated by a Control-Z. The "Q" instructs DED to Quit.]

U: I

U: This is a test line.<cr><lf>

S: I<cr><lf>\*<IAC><SB><RCTE><11><0><24><IAC><SE>

[DED prompts the user, during text input, with an asterisk at the beginning of every line. The server sets the break classes to classes 4 and 5, the format effectors and the non-format effector controls.]

P: I<cr><lf>\*This is a test line.

S: <cr><lf>\*<IAC><SB><RCTE><0><IAC><SE>

P: <cr><lf>\*This is another test line.

Remote Controlled Transmission and Echoing Telnet Option  
RFC 726, NIC 39237 (Mar. 8, 1977)

U: This is another test line.<↑Z>

U: Q

[Note that the "Q" will not immediately be printed on the terminal, since it must wait for authorization.]

S: ↑Z<cr><lf>:<IAC><SB><RCTE><15><1><IAC><255><IAC><SE>

[The returned "↑Z" is two characters, not the ASCII Control-Z or <sub>.

S: Q<cr><lf>@<IAC><SB><RCTE><11><1><24><IAC><SE>

P: Q<cr><lf>@

And the user is returned to the Exec level.



TELNET Output Line Width Option

1. Command name and code.

NAOL 8 (Negotiate About Output Line-width)

2. Command meanings

In the following, we are discussing a simplex connection, one half of a full duplex TELNET connection. On the simplex connection under discussion, by definition data passes from the data sender to the data receiver. If we consider the example of a computer transmitting data over a connection to a terminal where the data is printed, then the computer is the data sender and the terminal is the data receiver. Continuing to use this example, the NAOL option could be used to negotiate the line width to be used when printing lines from the computer on the terminal. To negotiate line width on the other half of the TELNET connection the parties involved reverse their data sender and data receiver roles; this can be done unambiguously as the sender of a DO or DON'T NAOL command can only be the data sender, thus defining the half of the TELNET connection under discussion, and the sender of a WILL or WON'T NAOL command can only be the data receiver.

IAC DO NAOL

The data sender requests or agrees to negotiate about output line width with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output line width considerations.

IAC DON'T NAOL

The data sender refused to negotiate about output line width with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOL

The data receiver requests or agrees to negotiate about output line width with the data sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output line width considerations.

---

Preceding page blank

IAC WON'T NAOL

The data receiver refuses to negotiate about output line width, or demands a return to the unnegotiated default mode.

IAC SB NAOL DS <8 bit value> IAC SE

The data sender specifies, with the 8 bit value, which party should handle output line width considerations and how. The code for DS is 1.

IAC SB NAOL DR <8 bit value> IAC SE

The data receiver specifies, with the 8 bit value, which party should handle output line width considerations and how. The code for DR is 0.

### 3. Default

DON'T NAOL

In the default absence of

WON'T NAOL

negotiation concerning which party, data sender or data receiver, is handling output line width considerations, neither party is required to handle line width consideration and neither party is prohibited from handling line width consideration but it is appropriate if at least the data receiver handles line width considerations albeit primitively.

### 4. Motivation for the Option

There appear to be four cases in which it is useful for the party at one end of a TELNET connection to communicate with the other party about the output line width:

- a) The sender may wish the receiver to use its local knowledge of the printer width to properly handle the line width;
- b) The receiver may wish the sender to use its local knowledge of the data being sent to properly handle the line width;
- c) The sender may wish to use its local knowledge of the data being sent to instruct the receiver in the proper handling of the line width; and

TELNET Output Line Width Option  
NIC 20196 (Nov. 13, 1973)

- d) The receiver may wish to use its local knowledge of the printer width to instruct the sender in the proper handling of the line width.

An example of proper handling of the line length is for the receiver to "fold" lines sent by the sender so that the lines fit on the printer page. Another example of proper handling of the line length might be not folding lines even though they overflow the printer page, as that is what the user desires.

## 5. Description of the Option

The data sender and data receiver use the 8 bit value along with the DS and DR SB commands as follows:

8 bit value	Meaning
0	command sender suggests he alone will handle output line-width considerations for the connection.
1 to 253	command sender suggests other party alone should handle output line-width considerations but suggests line width should be value given, in characters.
254	command sender suggests other party alone should handle output line-width considerations but suggests line width should be considered infinity.
255	command sender suggests other party alone should handle output line-width considerations and suggests nothing about how it should be done.

The guiding rules are that

- 1) if neither data receiver or data sender wants to handle output line-width considerations, the data receiver must do it, and
- 2) if both data receiver or data sender want to handle output line-width considerations, the data sender gets to do it.

TELNET Output Line Width Option  
NIC 20196 (Nov. 13, 1973)

The reasoning for the former rule is that if neither want to do it, then the default in the NAOL option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver makes.

6. Some sample negotiations are:

no subnegotiations

data receiver handles output line-width considerations

IAC SB NAOL DS 132 IAC SE  
IAC SB NAOL DR 0 IAC SE

data sender suggests data receiver handle output line-width consideration data with suggested line width of 132; receiver agrees.

IAC SB NAOL DR 255 IAC SE  
IAC SB NAOL DS 0 IAC SE

data receiver suggests data sender handle line-width considerations; sender refuses.

IAC SB NAOL DS 0 IAC SE  
IAC SB NAOL DR 72 IAC SE

data sender wants to handle line-width considerations; receiver agrees but notifies the sender the line printer only has 72 columns.

As with all option negotiation, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time either party can disable further negotiation by giving the appropriate WON'T NAOL or DON'T NAOL command.

TELNET Output Page Size Option  
NIC 20197 (Nov. 13, 1973)

TELNET Output Page Size Option

1. Command name and code.

NAOP 9 (Negotiate About Output Page-size)

(By page size we mean number of lines per page.)

2. Command meanings.

In the following, we are discussing a simplex connection, one half of a full duplex TELNET connection. On the simplex connection under discussion, by definition data passes from the data sender to the data receiver. If we consider the example of a computer transmitting data over a connection to a terminal where the data is printed, then the computer is the data sender and the terminal is the data receiver. Continuing to use this example, the NAOP option could be used to negotiate the page size to be used when printing pages from the computer on the terminal. To negotiate page size on the other half of the TELNET connection the parties involved reverse their data sender and data receiver roles; this can be done unambiguously as the sender of a DO or DON'T NAOP command can only be the data sender, thus defining the half of the TELNET connection under discussion, and the sender of a WILL or WON'T NAOP command can only be the data receiver.

IAC DO NAOP

The data sender requests or agrees to negotiate about output page size with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output page size considerations.

IAC DON'T NAOP

The data sender refuses to negotiate about output page size with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOP

The data receiver requests or agrees to negotiate about output page size with the data sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output page size considerations.

TELNET Output Page Size Option  
NIC 20197 (Nov. 13, 1973)

IAC WON'T NAOP

The data receiver refuses to negotiate about output page size, or demands a return to the unnegotiated default mode.

IAC SB NAOP DS <8 bit value> IAC SE

The data sender specifies, with the 8 bit value, which party should handle output page size considerations and how. The code for DS is 1.

IAC SB NAOP DR <8 bit value> IAC SE

The data receiver specifies, with the 8 bit value, which party should handle output page size considerations and how. The code for DR is 0.

### 3. Default

DON'T NAOP  
WON'T NAOP

In the default absence of negotiation concerning which party, data sender or data receiver, is handling output page size considerations, neither party is required to handle page size consideration and neither party is prohibited from handling page size consideration, but it is appropriate if at least the data receiver handles page size considerations albeit primitively.

### 4. Motivation for the Option

There appear to be four cases in which it is useful for the party at one end of a TELNET connection to communicate with the other party about the output page size:

- a) the sender may wish the receiver to use its local knowledge of the printer page size to properly handle the page size;
- b) the receiver may wish the sender to use its local knowledge to the data being sent to properly handle the page size;
- c) the sender may wish to use its local knowledge of the data being sent to instruct the receiver in the proper handling of the page size; and

TELNET Output Page Size Option  
NIC 20197 (Nov. 13, 1973)

- d) the receiver may wish to use its local knowledge of the printer size to instruct the sender in the proper handling of the page size.

An example of proper handling of the page size is for the receiver to hold off further output until instructed to continue when the lines being printed are about to overflow the scope face.

5. Description of the Option.

The data sender and data receiver use the 8 bit value along with the DS and DR SB commands as follows.

8 bit value	Meaning
0	command sender suggests he alone will handle output page-size considerations for the connection.
1 to 253	command sender suggests other party alone should handle output page-size considerations but suggests page size should be value given, in lines.
254	command sender suggests other party alone should handle output page size considerations but suggests page size should be considered infinity.
255	command sender suggests other party alone should handle output page size considerations and suggests nothing about how it should be done.

The guiding rules are that

- 1) if neither data receiver or data sender wants to handle output page size considerations, the data receiver must do it, and
- 2) if both data receiver or data sender want to handle output page size, the data sender gets to do it.

The reasoning for the former rule is that if neither want to do it, then the default in the NAOP option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver makes.

TELNET Output Page Size Option  
NIC 20197 (Nov. 13, 1973)

Some sample negotiations are:

no subnegotiations

data receiver handles output page size considerations

IAC SB NAOP DS 66 IAC SE  
IAC SB NAOP DR 0 IAC SE

data sender suggests data receiver handle output page size consideration data with suggested page size of 66 lines; receiver agrees.

IAC SB NAOP DR 255 IAC SE  
IAC SB NAOP DS 0 IAC SE

data receiver suggests data sender handle page size condiseration; sender refuses.

IAC SB NAOP DS 0 IAC SE  
IAC SB NAOP DR 30 IAC SE

data sender wants to handle page size considerations; receiver agrees but notifies the sender the scope only has 30 IAC SE lines.

As with all option negotiation, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time either party can disable further negotiation by giving the appropriate WON'T NAOP or DON'T NAOP command.

TELNET OUTPUT CARRIAGE-RETURN DISPOSITION OPTION  
RFC 652, NIC 31155 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 652, NIC 31155 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>NAOCRD.TXT

TELNET OUTPUT CARRIAGE-RETURN DISPOSITION OPTION

1. Command name and code

NAOCRD 10 (Negotiate About Output Carriage-Return Disposition)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet options.

IAC DO NAOCRD

The data sender requests or agrees to negotiate about output carriage-return character disposition with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output carriage-returns.

IAC DON'T NAOCRD

The data sender refuses to negotiate about output carriage-return disposition with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOCRD

The data receiver requests or agrees to negotiate about output carriage-return disposition with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output carriage-returns.

IAC WON'T NAOCRD

The data receiver refuses to negotiate about output carriage-return disposition, or demands a return to the unnegotiated default mode.

TELNET OUTPUT CARRIAGE-RETURN DISPOSITION OPTION  
RFC 652, NIC 31155 (Oct. 25, 1974)

IAC SB NAOCRD DS <8-bit value> IAC SE

The data sender specifies, with the 8-bit value, which party should handle carriage-returns and what their disposition should be. The code for DS is 1.

IAC SB NAOCRD DR <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value, which party should handle carriage-returns and what their disposition should be. The code for DR is 0.

3. Default

DON'T NAOCRD/WON'T NAOCRD.

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output carriage-returns, neither party is required to handle carriage-returns and neither party is prohibited from handling them; but it is appropriate if at least the data receiver handles carriage-returns, albeit primitively.

4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOP Telnet option descriptions.

5. Description of the Option

The data sender and the data receiver use the 8-bit value along with the NAOCRD SB commands as follows:

8-bit value	Meaning
0	Command sender suggests that he alone will handle carriage-returns, for the connection.
1 to 250	Command sender suggests that the other party alone should handle carriage-returns, but suggests that a delay of the indicated value be used. The value is the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. (See qualification, below.)
251	Not allowed, in order to be compatible with related Telnet options.

TELNET OUTPUT CARRIAGE-RETURN DISPOSITION OPTION  
RFC 652, NIC 31155 (Oct. 25, 1974)

252 Command sender suggests that the other party alone handle carriage-returns, but suggests that they be discarded.  
253 Not allowed, in order to be compatible with related Telnet options.  
254 Command sender suggests that the other party alone should handle carriage-returns but suggests waiting for a character to be transmitted (on the other simplex connection) before sending more data. (See qualification, below.) Note that, due to the assynchrony of the two simplex connections, phase problems can occur with this option.  
255 Command sender suggests that the other party alone should handle carriage-returns and suggests nothing about how it should be done.

The guiding rules are that:

- 1) if neither data receiver nor data sender wants to handle carriage-returns, the data receiver must do it, and
- 2) if both data receiver and data sender want to handle carriage-returns, the data sender gets to do it.

The reasoning for the former rule is that if neither wants to do it, then the default in the NAOCRD option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

Note that carriage-return delays, controlled by the data sender, must consist of NUL characters inserted immediately after the character in question. This is necessary due to the assynchrony of network transmissions. Due to the Telnet end-of-line convention, with carriage-returns followed by a linefeed, any NULs that would otherwise be placed after the carriage-return must be placed after the linefeed, regardless of any modifications that may additionally be made to the line feed (see NAOLFD Telnet option).

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOCRD or DON'T NAOCRD command.



TELNET Output Horizontal Tabstops Option  
RFC 653, NIC 31156 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 653, NIC #31156 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>NAOHTS.TXT

TELNET Output Horizontal Tabstops Option

1. Command name and code

NAOHTS 11 (Negotiate About Output Horizontal Tabstops)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet options.

IAC DO NAOHTS

The data sender requests or agrees to negotiate about output horizontal tabstops with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output horizontal tabstops.

IAC DON'T NAOHTS

The data sender refuses to negotiate about output horizontal tabstops with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOHTS

The data receiver requests or agrees to negotiate about output horizontal tabstops with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output horizontal tabstops.

IAC WON'T NAOHTS

The data receiver refuses to negotiate about output horizontal tabstops, or demands a return to the unnegotiated default mode.

---

Preceding page blank

TELNET Output Horizontal Tabstops Option  
RFC 653, NIC 31156 (Oct. 25, 1974)

IAC SB NAOHTS DS <8-bit value> ... <8-bit value> IAC SE

The data sender specifies, with the 8-bit value(s), which party should handle output horizontal tabstop considerations and what the stops should be. The code for DS is 1.

IAC SB NAOHTS DR <8-bit value> ... <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value(s), which party should handle output horizontal tabstop considerations and what the stops should be. The code for DR is 0.

### 3. Default

DON'T NAOHTS/WON'T NAOHTS.

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output horizontal tabstops, neither party is required to handle them and neither party is prohibited from handling them; but it is appropriate if at least the data receiver handles horizontal tabstops, albeit primitively.

### 4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOP Telnet option descriptions.

### 5. Description of the Option

The data sender and the data receiver use the 8-bit value(s) along with the DS and DR SB subcommands as follows (multiple 8-bit values are allowed only if each is greater than zero and less than 251):

8-bit value	Meaning
0	Command sender suggests that he alone will handle tabstops, for the connection.
1 to 250	Command sender suggests that the other party alone should handle tabstop considerations, but suggests that the indicated value(s) be used. The value(s) are the column numbers, relative to the physical left side of the printer page or terminal screen, that are to be set.
251 to 254	Not allowed, in order to be compatible with related Telnet options.

TELNET Output Horizontal Tabstops Option  
RFC 653, NIC 31156 (Oct. 25, 1974)

255           Command sender suggests that the other party alone should handle output tabstops and suggests nothing about how it should be done.

The guiding rules are that:

- (1) if neither data receiver nor data sender wants to handle output horizontal tabstops, the data receiver must do it, and
- (2) if both data receiver and data sender want to handle output horizontal tabstops, the data sender gets to do it.

The reasoning for the former rule is that if neither wants to do it, then the default in the NAOHTS option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOHTS or DON'T NAOHTS command.



TELNET Output Horizontal Tab Disposition Option  
RFC 654, NIC 31157 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 654, NIC 31157 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>NAOHTD.TXT

TELNET Output Horizontal Tab Disposition Option

1. Command name and code

NAOHTD 12

(Negotiate About Output Horizontal Tab Disposition)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet options.

IAC DO NAOHTD

The data sender requests or agrees to negotiate about output horizontal tab character disposition with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output horizontal tab character considerations.

IAC DON'T NAOHTD

The data sender refuses to negotiate about output horizontal tab characters with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOHTD

The data receiver requests or agrees to negotiate about output horizontal tab characters with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output horizontal tab character considerations.

IAC WON'T NAOHTD

The data receiver refuses to negotiate about output horizontal tab characters, or demands a return to the unnegotiated default mode.

---

Preceding page blank

TELNET Output Horizontal Tab Disposition Option  
RFC 654, NIC 31157 (Oct. 25, 1974)

IAC SB NAOHTD DS <8-bit value> IAC SE

The data sender specifies, with the 8-bit value, which party should handle output horizontal tab characters and what their disposition should be. The code for DS is 1.

IAC SB NAOHTD DR <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value, which party should handle output horizontal tab characters and what their disposition should be. The code for DR is 0.

### 3. Default

DON'T NAOHTD/WON'T NAOHTD.

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output horizontal tab character considerations, neither party is required to handle horizontal tab characters and neither party is prohibited from handling them; but it is appropriate if at least the data receiver handles horizontal tab character considerations, albeit primitively.

### 4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOP Telnet option descriptions.

### 5. Description of the Option

The data sender and the data receiver use the 8-bit value along with the DS and DR SB commands as follows:

8-bit value	Meaning
0	Command sender suggests that he alone will handle horizontal tab characters, for the connection.
1 to 250	Command sender suggests that the other party alone should handle horizontal tab characters, but suggests that a delay of the indicated value be used. The value is the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character.

TELNET Output Horizontal Tab Disposition Option  
RFC 654, NIC 31157 (Oct. 25, 1974)

- 251 Command sender suggests that the other party alone handle horizontal tabs, but suggests that each occurrence of the character be replaced by a space.
- 252 Command sender suggests that the other party alone handle horizontal tabs, but suggests that they be discarded.
- 253 Command sender suggests that the other party alone should handle horizontal tab characters, but suggests that tabbing be simulated.
- 254 Command sender suggests that the other party alone should handle horizontal tab characters, but suggests that waiting for a character to be transmitted (on the other simplex connection) before sending more data. Note that, due to the assynchrony of the two simplex connections, phase problems can occur with this option.
- 255 Command sender suggests that the other party alone should handle output horizontal tabs and suggests nothing about how it should be done.

The guiding rules are that:

- 1) if neither data receiver nor data sender wants to handle output horizontal tab characters, the data receiver must do it, and
- 2) if both data receiver and data sender wants to handle output horizontal tab characters, the data sender gets to do it.

The reasoning for the former rule is that if neither wants to do it, then the default in the NAOHTD option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

Simulation is defined as the replacement of the horizontal tab character by enough spaces to move the printer head (or line-pointer) to the next horizontal tab stop.

Note that delays, controlled by the data sender, must consist of NUL characters inserted immediately after the horizontal tab character. This is necessary due to the assynchrony of network transmissions.

TELNET Output Horizontal Tab Disposition Option  
RFC 654, NIC 31157 (Oct. 25, 1974)

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOHTD or DON'T NAOHTD command.

TELNET Output Formfeed Disposition Option  
RFC 655 NIC 31158 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 655, NIC 31158 (Oct. 25, 74)  
Online file: [ISI]<DCROCKER>NAOFFD.TXT

TELNET Output Formfeed Disposition Option

1. Command name and code

NAOFFD - 13

(Negotiate About Output Formfeed Disposition)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet Options specifications.

IAC DO NAOFFD

The data sender requests or agrees to negotiate about output formfeed disposition with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output formfeeds.

IAC DON'T NAOFFD

The data sender refuses to negotiate about output formfeed disposition with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOFFD

The data receiver requests or agrees to negotiate about output formfeed disposition with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output formfeeds.

IAC WON'T NAOFFD

The data receiver refuses to negotiate about output formfeed disposition, or demands a return to the unnegotiated default mode.

TELNET Output Formfeed Disposition Option  
RFC 655 NIC 31158 (Oct. 25, 1974)

IAC SB NAOFFD DS <8-bit value> IAC SE

The data sender specifies, with the 8-bit value, which party should handle formfeeds and what their disposition should be. The code for DS is 1.

IAC SB NAOFFD DR <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value, which party should handle formfeeds and what their disposition should be. The code for DR is 0.

### 3. Default

DON'T NAOFFD/WON'T NAOFFD

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output formfeeds, neither party is required to handle formfeeds and neither party is prohibited from handling them; but it is appropriate if at least the data receiver handles formfeed considerations, albeit primitively.

### 4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOFFD Telnet option descriptions.

### 5. Description of the Option

The data sender and the data receiver use the 8-bit value along with the DS and DR SB commands as follows:

8-bit value	Meaning
0	Command sender suggests that he alone will handle formfeeds, for the connection.
1 to 250	Command sender suggests that the other party alone should handle formfeeds, but suggests that the indicated value be used. The value is the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character.

TELNET Output Formfeed Disposition Option  
RFC 655 NIC 31158 (Oct. 25, 1974)

- 251 Command sender suggests that the other party alone handle formfeeds, but suggests that each occurrence of the character be replaced by carriage-return/line-feed.
- 252 Command sender suggests that the other party alone handle formfeeds, but suggests that they be discarded.
- 253 Command sender suggests that the other party alone should handle formfeeds, but suggests that formfeeds be simulated.
- 254 Command sender suggests that the other party alone should handle output formfeeds but suggests waiting for a character to be transmitted (on the other simplex connection) before sending more data. Note that, due to the assynchrony of the two simplex connections, phase problems can occur with this option.
- 255 Command sender suggests that the other party alone should handle output formfeeds and suggests nothing about how it should be done.

The guiding rules are that:

- 1) if neither data receiver nor data sender wants to handle output formfeeds, the data receiver must do it, and
- 2) if both data receiver and data sender want to handle output formfeeds, the data sender gets to do it.

The reasoning for the former rule is that if neither wants to do it, then the default in the NAOFFD option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

Simulation is defined as the replacement of the formfeed character by enough line-feeds (only) to advance the paper (or line-pointer) to the top of the next page (or to the top of the terminal screen).

Note that delays, controlled by the data sender, must consist of NUL characters inserted immediately after the formfeed character. This is necessary due to the assynchrony of network transmission.

TELNET Output Formfeed Disposition Option  
RFC 655 NIC 31158 (Oct. 25, 1974)

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOFFD or DON'T NAOFFD command.

TELNET Output Vertical Tabstops Option  
RFC 656, NIC 31159 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 656, NIC 31159 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>NAOVTS.TXT

TELNET Output Vertical Tabstops Option

1. Command name and code

NAOVTS 14

(Negotiate About Vertical Tabstops)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet Options specifications.

IAC DO NAOVTS

The data sender requests or agrees to negotiate about output vertical tabstops with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output vertical tabstop considerations.

IAC DON'T NAOVTS

The data sender refuses to negotiate about output vertical tabstops with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOVTS

The data receiver requests or agrees to negotiate about output vertical tabstops with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output vertical tabstop considerations.

IAC WON'T NAOVTS

The data receiver refuses to negotiate about output vertical tabstops, or demands a return to the unnegotiated default mode.

TELNET Output Vertical Tabstops Option  
RFC 656, NIC 31159 (Oct. 25, 1974)

IAC SB NAOVTS DS <8-bit value> ... <8-bit value> IAC SE

The data sender specifies, with the 8-bit value(s), which party should handle output vertical tabstop considerations and what the stops should be. The code for DS is 1.

IAC SB NAOVTS DR <8-bit value> ... <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value(s), which party should handle output vertical tabstop considerations and what the stops should be. The code for DR is 0.

3. Default

DON'T NAOVTS/WON'T NAOVTS.

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output vertical tabstop considerations, neither party is required to handle vertical tabstops and neither party is prohibited from handling them; but it is appropriate if at least the data receiver handles vertical tabstop considerations, albeit primitively.

4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOVTS Telnet option descriptions.

5. Description of the Option

The data sender and the data receiver use the 8-bit value(s) along with the DS and DR SB commands as follows (multiple 8-bit values are allowed only if each is greater than zero and less than 251):

8-bit value	Meaning
0	Command sender suggests that he alone will handle the vertical tabstops, for the connection.
1 to 250	Command sender suggests that the other party alone should handle the stops, but suggests that the indicated value(s) be used. Each value is the line number, relative to the top of the printer page or terminal screen, that is to be set as a vertical tabstop.

TELNET Output Vertical Tabstops Option  
RFC 656, NIC 31159 (Oct. 25, 1974)

- 251 to 254 Not allowed, in order to be compatible with related Telnet options.
- 255 Command sender suggests that the other party alone should handle output vertical tabstops and suggests nothing about how it should be done.

The guiding rules are that:

- 1) if neither data receiver nor data sender wants to handle output vertical tabstops, the data receiver must do it, and
- 2) if both data receiver and data sender want to handle output vertical tabstops, the data sender gets to do it.

The reasoning for the former rule is that if neither wants to do it, then the default in the NAOVTS option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make. This is necessary due to the assynchrony of network transmissions.

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOVTS or DON'T NAOVTS command.



TELNET Output Vertical Tab Disposition Option  
RFC 657, NIC 31160 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 657, NIC 31160 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>NAOVTD.TXT

TELNET Output Vertical Tab Disposition Option

1. Command name and code

NAOVTD 15

(Negotiate About Output Vertical Tab Disposition)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet Options specifications.

IAC DO NAOVTD

The data sender requests or agrees to negotiate about output vertical tab character disposition with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output vertical tab character considerations.

IAC DON'T NAOVTD

The data sender refuses to negotiate about output vertical tab character disposition with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOVTD

The data receiver requests or agrees to negotiate about output vertical tab character disposition with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output vertical tab character considerations.

IAC WON'T NAOVTD

The data receiver refuses to negotiate about output vertical tab character disposition, or demands a return to the unnegotiated default mode.

---

Preceding page blank

TELNET Output Vertical Tab Disposition Option  
RFC 657, NIC 31160 (Oct. 25, 1974)

IAC SB NAOVTD DS <8-bit value> IAC SE

The data sender specifies, with the 8-bit value, which party should handle output vertical tab characters and what their disposition should be. The code for DS is 1.

IAC SB NAOVTD DR <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value, which party should handle output vertical tab characters and what their disposition should be. The code for DR is 0.

### 3. Default

DON'T NAOVTD/WON'T NAOVTD

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output vertical tab character considerations, neither party is required to handle vertical tab characters and neither party is prohibited from handling them; but it is appropriate if at least the data receiver handles vertical tab character considerations, albeit primitively.

### 4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOVTD Telnet option descriptions.

### 5. Description of the Option

The data sender and the data receiver use the 8-bit value along with the DS and DR SB commands as follows:

8-bit value	Meaning
0	Command sender suggests that he alone will handle vertical tab characters, for the connection.
1 to 250	Command sender suggests that the other party alone should handle tab characters, but suggests that a delay of the indicated value be used. The value is the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character.

TELNET Output Vertical Tab Disposition Option  
RFC 657, NIC 31160 (Oct. 25, 1974)

- 251 Command sender suggests that the other party alone handle vertical tabs, but suggests that each occurrence of the character be replaced by carriage-return/linefeed.
- 252 Command sender suggests that the other party alone handle vertical tabs, but suggests that they be discarded.
- 253 Command sender suggests that the other party alone should handle tab characters, but suggests that tabbing be simulated.
- 254 Command sender suggests that the other party alone should handle the output disposition but suggests waiting for a character to be transmitted (on the other simplex connection) before sending more data. Note that, due to the assynchrony of the two simplex connections, phase problems can occur with this option.
- 255 Command sender suggests that the other party alone should handle the output disposition and suggests nothing about how it should be done.

The guiding rules are that:

- 1) if neither data receiver nor data sender wants to handle the output vertical tab characters, the data receiver must do it, and
- 2) if both data receiver and data sender want to handle the output vertical tab characters, the data sender gets to do it.

The reasoning for the former rule is that if neither want to do it, then the default in the NAOVTD option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

Simulation is defined as the replacement of the character by enough line-feeds (only) to advance the paper (or line-pointer) to the next vertical tab stop.

Note that delays, controlled by the data sender, must consist of NUL characters, inserted immediately after the line-feed character. This is necessary due to the assynchrony of network transmissions.

TELNET Output Vertical Tab Disposition Option  
RFC 657, NIC 31160 (Oct. 25, 1974)

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOVTD or DON'T NAOVTD command.

TELNET OUTPUT LINEFEED DISPOSITION  
RFC 658, NIC 31161 (Oct. 25, 1974)

D. Crocker (UCLA-NMC)  
RFC 658, NIC 31161 (Oct. 25, 1974)  
Online file: [ISI]<DCROCKER>NAOLFD.TXT

TELNET OUTPUT LINEFEED DISPOSITION

1. Command name and code

NAOLFD 16

(Negotiate About Output Linefeed Disposition)

2. Command meanings

In the following, we are discussing a simplex connection, as described in the NAOL and NAOP Telnet Options.

IAC DO NAOLFD

The data sender requests or agrees to negotiate about output linefeed disposition with the data receiver. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver is assumed to be handling output linefeed considerations.

IAC DON'T NAOLFD

The data sender refuses to negotiate about output linefeed disposition with the data receiver, or demands a return to the unnegotiated default mode.

IAC WILL NAOLFD

The data receiver requests or agrees to negotiate about output linefeed disposition with the sender. In the case where agreement has been reached and in the absence of further subnegotiations, the data receiver alone is assumed to be handling output linefeed considerations.

IAC WON'T NAOLFD

The data receiver refuses to negotiate about output linefeed disposition, or demands a return to the unnegotiated default mode.

TELNET OUTPUT LINEFEED DISPOSITION  
RFC 658, NIC 31161 (Oct. 25, 1974)

IAC SB NAOLFD DS <8-bit value> IAC SE

The data sender specifies, with the 8-bit value, which party should handle output linefeeds and what their disposition should be. The code for DS is 1.

IAC SB NAOLFD DR <8-bit value> IAC SE

The data receiver specifies, with the 8-bit value, which party should handle output linefeeds and what their disposition should be. The code for DR is 0.

3. Default

DON'T NAOLFD/WON'T NAOLFD.

In the default absence of negotiations concerning which party, data sender or data receiver, is handling output linefeed considerations, neither party is required nor prohibited from handling linefeeds; but it is appropriate if at least the data receiver handles them, albeit primitively.

4. Motivation for the Option

Please refer to section 4 of the NAOL and of the NAOLFD Telnet option descriptions.

5. Description of the Option

The data sender and the data receiver use the 8-bit value along with DS and DR SB commands as follows:

8-bit value	Meaning
0	Command sender suggests that he alone will handle linefeeds, for the connection.
1 to 250	Command sender suggests that the other party alone should handle linefeeds, but suggests that a delay of the indicated value be used. The value is the number of character-times to wait or number of NULs to insert in the data stream before sending the next data character. (See qualifications, below.)
251	Not allowed, in order to be compatible with related Telnet options.

TELNET OUTPUT LINEFEED DISPOSITION  
RFC 658, NIC 31161 (Oct. 25, 1974)

- 252      Command sender suggests that the other party alone handle linefeeds, but suggests that they be discarded.
- 253      Command sender suggests that the other party alone should handle linefeeds, but suggests that linefeeds be simulated.
- 254      Command sender suggests that the other party alone should handle output linefeeds but suggests waiting for a character to be transmitted (on the other simplex connection) before sending more data. (See qualifications, below.) Note that, due to the assynchrony of the two simplex connections, phase problems can occur with this option.
- 255      Command sender suggests that the other party alone should handle output linefeeds and suggests nothing about how it should be done.

The guiding rules are that:

- 1) if neither data receiver nor data sender wants to handle output linefeeds, the data receiver must do it, and
- 2) if both data receiver and data sender want to handle output linefeed disposition, the data sender gets to do it.

The reasoning for the former rule is that if neither wants to do it, then the default in the NAOLFD option dominates. If both want to do it, the sender, who is presumed to have special knowledge about the data, should be allowed to do it, taking into account any suggestions the receiver may make.

Simulation is defined as the replacement of the linefeed character by new-line (see following) and enough blanks to move the print head (or line pointer) to the same lateral position it occupied just prior to receiving the linefeed. To avoid infinite recursion, such simulation is allowed only for linefeed characters that are not immediately preceded by carriage-returns (that is, part of a Telnet new-line combination). It is assumed that linefeed simulation will be necessary for printers that do not have a separate linefeed (like the IBM 2741); in this case, end-of-line character padding can be specified through NAOCRD. Any padding ( $0 < \text{<8-bit-value>} < 251$ ) of linefeed characters is to be done for ALL linefeed characters.

TELNET OUTPUT LINEFEED DISPOSITION  
RFC 658, NIC 31161 (Oct. 25, 1974)

NOTE: Delays, controlled by the data sender, must consist of NUL characters inserted immediately after the character. This is necessary due to the assynchrony of network transmissions. Additionally, due to the presence of the Telnet end-of-line convention, it may be necessary to add carriage-return padding or delay after the associated linefeed (see NAOCRD Telnet option).

As with all option negotiations, neither party should suggest a state already in effect except to refuse to negotiate; changes should be acknowledged; and once refused, an option should not be resuggested until "something changes" (e.g., another process starts).

At any time, either party can disable further negotiation by giving the appropriate WON'T NAOLFD or DON'T NAOLFD command.

TELNET EXTENDED ASCII OPTION  
RFC 698, NIC 32964 (July 23, 1975)

Tovar Mock (SU-AI)  
RFC 698, NIC 32964 (July 23, 1975)

TELNET EXTENDED ASCII OPTION

1. Command Name and code.

EXTEND-ASCII 17

2. Command Meanings.

IAC WILL EXTEND-ASCII

The sender of this command requests permission to begin transmitting, or confirms that it may now begin transmitting extended ASCII, where additional 'control' bits are added to normal ASCII, which are treated specially by certain programs on the host computer.

IAC WON'T EXTEND-ASCII

If the connection is already being operated in extended ASCII mode, the sender of this command demands that the receiver begin transmitting data characters in standard NVT ASCII. If the connection is not already being operated in extended ASCII mode, The sender of this command refuses to begin transmitting extended ASCII.

IAC DO EXTEND-ASCII

The sender of this command requests that the receiver begin transmitting, or confirms that the receiver of this command is allowed to begin transmitting extended ASCII.

IAC DON'T EXTEND-ASCII

The sender of this command demands that the receiver of this command stop or not start transmitting data in extended ASCII mode.

IAC SB EXTASC  
<high order bits (bits 15-8)><low order bits (bits 7-0)> IAC SE

This command transmits an extended ASCII character in the form of two 8-bit bytes. Each 8-bit byte contains 8 data bits.

TELNET EXTENDED ASCII OPTION  
RFC 698, NIC 32964 (July 23, 1975)

3. Default

DON'T EXTEND-ASCII

WON'T EXTEND-ASCII

i.e., only use standard NVT ASCII

4. Motivation.

Several sites on the net, for example, SU-AI and MIT-AI, use keyboards which use almost all 128 characters as printable characters, and use one or more additional bits as 'control' bits as command modifiers or to separate textual input from command input to programs. Without these additional bits, several characters cannot be entered as text because they are used for control purposes, such as the greek letter 'beta' which on a TELNET connection is CONTROL-C and is used for stopping ones job. In addition there are several commonly used programs at these sites which require these additional bits to be run effectively. Hence it is necessary to provide some means of sending characters larger than 8 bits wide.

5. Description of the option.

This option is to allow the transmission of extended ASCII.

Experience has shown that most of the time, 7-bit ASCII is typed, with an occasional 'control' character used. Hence, it is expected normal NVT ASCII would be used for 7-bit ASCII and that extended-ASCII be sent as an escape character sequence.

The exact meaning of these additional bits depends on the user program. At SU-AI and at MIT-AI, the first two bits beyond the normal 7-bit ASCII are passed on to the user program and are denoted as follows.

Bit 8 (or 200 octal) is the CONTROL bit  
Bit 9 (or 400 octal) is the META bit

(NOTE: 'CONTROL' is used in a non-standard way here; that is, it usually refers to codes 0-37 in NVT ASCII. CONTROL and META are echoed by prefixing the normal character with 013 (integral symbol) for CONTROL and 014 (plus-minus) for META. If both are present, it is known as CONTROL-META and echoed as 013 014 7-bit character.)

TELNET EXTENDED ASCII OPTION  
RFC 698, NIC 32964 (July 23, 1975)

6. Description of Stanford Extended ASCII Characters

In this section, the extended graphic character set used at SU-AI is described for reference, although this specific character set is not required as part of the extended ASCII Telnet option. Characters described as "hidden" are alternate graphic interpretations of codes normally used as format effectors, used by certain typesetting programs.

Code	Graphic represented
000	null (hidden vertically centered dot)
001	downward arrow
002	alpha (all Greek letters are lowercase)
003	beta
004	logical and (caret)
005	logical not (dash with downward extension)
006	epsilon
007	pi
010	lambda
011	tab (hidden gamma)
012	linefeed (hidden delta)
013	vertical tab (hidden integral)
014	formfeed (hidden plus-minus)
015	carriage return (hidden circled-plus)
016	infinity
017	del (partial differential)
020	proper subset (right-opening horseshoe)
021	proper superset (left-opening horseshoe)
022	intersection (down-opening horseshoe)
023	union (up-opening horseshoe)
024	universal quantifier (upside-down A)
025	existential quantifier (backwards E)
026	circled-times
027	left-right double headed arrow
030	underbar
031	right pointing arrow
032	tilde
033	not-equal
034	less-than-or-equal
035	greater-than-or-equal
036	equivalence (column of 3 horizontal bars)
037	logical or (V shape)
040-135	as in standard ASCII

TELNET EXTENDED ASCII OPTION  
RFC 698, NIC 32964 (July 23, 1975)

136	upward pointing arrow
137	left pointing arrow
140-174	as in standard ASCII
175	altmode (prints as lozenge)
176	right brace
177	rubout (hidden circumflex)

TELNET Logout Option  
RFC 727, NIC 40025 (Apr. 27, 1977)

Mark Crispin (MIT-AI)  
RFC 727, NIC 40025 (Apr. 27, 1977)

TELNET Logout Option

1. Command name and code.

LOGOUT 18

2. Command meanings.

IAC WILL LOGOUT

The sender of this command REQUESTS permission to, or confirms that it will, forcibly log off the user process at its end.

IAC WON'T LOGOUT

The sender of this command REFUSES to forcibly log off the user process at its end.

IAC DO LOGOUT

The sender of this command REQUESTS that the receiver forcibly log off the user process at the receiver's end, or confirms that the receiver has its permission to do so.

IAC DON'T LOGOUT

The sender of this command DEMANDS that the receiver not forcibly log off the user process at the receiver's end.

3. Default.

WON'T LOGOUT

DON'T LOGOUT

i.e., no forcible logging off of the server's user process.

#### 4. Motivation for the option.

Often, a runaway user process can be hung in such a state that it cannot be interrupted by normal means. Conversely, the system itself can be bottlenecked so that response delays are intolerable. A user (human or otherwise) eventually will time out of frustration and take the drastic means of closing the connection to free itself from the hung process. In some situations, even the simple operation of logging out can take a long time.

Some systems treat a close to mean that it should log out its user process under it. However, many hosts merely "detach" the process so that an accidental close due to a user or temporary hardware error will not cause all work done on that job to be lost; when the connection is re-established, the user may "attach" back to its process. While this protection is often valuable, if the user is giving up completely on the host, it can cause this hung job to continue to load the system.

This option allows a process to instruct the server that the user process at the server's end should be forcibly logged out instead of detached. A secondary usage of this option might be for a server to warn of impending auto-logout of its user process due to inactivity.

#### 5. Description of the option.

When a user decides that it no longer wants its process on the server host and decides that it does not want to wait until the host's normal log out protocol has been gone through, it sends IAC DO LOGOUT. The receiver of the command may respond with IAC WILL LOGOUT, in which case it will then forcibly log off the user process at its end. If it responds with IAC WON'T LOGOUT, then it indicates that it has not logged off the user process at its end, and if the connection is broken, the process very possibly will be detached.

A truly impatient user that feels that it must break away from the server immediately could even send IAC DO LOGOUT and then close. At the worst, the server would only ignore the request and detach the user process. A server that implements the LOGOUT option should know to log out the user process despite the sudden close and even an inability to confirm the LOGOUT request!

#### 6. A sample implementation of the option.

The server implements the LOGOUT option both for accepting LOGOUT requests and for auto-logout warning.

Case 1:

The user connects to the server, and starts interacting with the server. For some reason, the user wishes to terminate interaction with the server, and is reluctant to go through the normal log out procedure, or perhaps the user is unable to go through the normal log out procedure. It does not want the process at the server any more, so it sends IAC DO LOGOUT. The server verifies the request with IAC WILL LOGOUT, and then forcibly logs off the user process (perhaps by using a system call that causes another process to be logged out). It does not have to close the connection unless the user closes or it wants to close. Neither does it wait until the user has received its confirmation--it starts the log out immediately so if the user has in the mean time closed the connection without waiting for confirmation, its logout request still is performed.

Case 2:

The user connects to the server, and after logging in, is idle for a while, long enough to approach the server's autologout time. The server shortly before the autologout sends IAC WILL LOGOUT; the user sees this and sends IAC DON'T LOGOUT, and continues work on the host. Nothing prevents the server from logging out the user process if inactivity continues; this can be used to prevent a malicious user from locking up a process on the server host by the simple expedient of sending IAC DON'T LOGOUT every time it sees IAC WILL LOGOUT but doing nothing else.



TELNET BYTE MACRO OPTION  
RFC 735, NIC 42083 (Nov. 3, 1977)

David H. Crocker (Crocker@RAND-UNIX)  
Rand-ISD  
Richard H. Gumpertz (Gumpertz@CMU-10A)  
Carnegie-Mellon University

RFC 735, NIC 42083 (Nov. 3, 1977)  
Obsoletes RFC 729 (NIC 40306)

REVISED TELNET BYTE MACRO OPTION

1. Command name and code:

BM 19

2. Command Meanings:

IAC WILL BM

The sender of this command REQUESTS or AGREES to use the BM option, and will send single data characters which are to be interpreted as if replacement data strings had been sent.

IAC WON'T BM

The sender of this option REFUSES to send single data characters which are to be interpreted as if replacement data strings had been sent. Any existing BM <macro byte> definitions are discarded (i.e., reset to their original data interpretations).

IAC DO BM

The sender REQUESTS or AGREES to have the other side (sender of WILL BM) send single data characters which are to be interpreted as if replacement data strings had been sent.

IAC DON'T BM

The sender REFUSES to allow the other side to send single data characters which are to be interpreted as if replacement data strings had been sent. Any existing BM <macro byte> definitions are to be discarded.

Preceding page blank

TELNET BYTE MACRO OPTION  
RFC 735, NIC 42083 (Nov. 3, 1977)

IAC SB BM <DEFINE> <macro byte> <count>  
  <replacement string> IAC SE

where:

<macro byte> is the data byte actually to be sent across the network; it may NOT be Telnet IAC (decimal 255, but may be any other 8-bit character).

<count> is one 8-bit byte binary number, indicating how many <replacement string> characters follow, up to the ending IAC SE, but not including it. NOTE: that doubled IACs in the definition should only be counted as one character per pair.

<replacement string> is a string of zero or more Telnet ASCII characters and/or commands, which the <macro byte> is to represent; any character may occur within a <replacement string>. Note, however, that an IAC in the string must be doubled, to be interpreted later as an IAC; to be interpreted later as data byte 255, it must be quadrupled in the original <replacement string> specification.

The indicated <macro byte> will be sent instead of the indicated <replacement string>. The receiver of the <macro byte> (the sender of the DO BM) is to behave EXACTLY as if the <replacement string> string of bytes had instead been received from the network. This interpretation is to occur before any other Telnet interpretations, unless the <macro byte> occurs as part of a Telnet command; in this case no special interpretation is to be made. In particular, an entire Telnet subnegotiation (i.e. from IAC SB through IAC SE) is to be considered a Telnet command in which NO replacement should be done.

The effect of a particular <macro byte> may be negated by resetting it to "expand" into itself.

IAC SB BM <DEFINE> X <0> IAC SE may be used to cause X to be ignored in the data stream.

<DEFINE> is decimal 1.

TELNET BYTE MACRO OPTION  
RFC 735, NIC 42083 (Nov. 3, 1977)

IAC SB BM <ACCEPT> <macro byte> IAC SE

The receiver of the <DEFINE> for <macro byte> accepts the requested definition and will perform the indicated replacement whenever a <macro byte> is received and is not part of any IAC Telnet command sequence.

<ACCEPT> is decimal 2.

IAC SB BM <REFUSE> <macro byte> <REASON> IAC SE

The receiver of the <DEFINE> for <macro byte> refuses to perform the indicated translation from <macro byte> to <replacement string> because the particular <macro byte> is not an acceptable choice, the length of the <replacement string> exceeds available storage, the length of the actual <replacement string> did not match the length predicted in the <count>, or for some unspecified reason.

<REFUSE> is decimal 3.

<REASON> may be

- |                |   |
|----------------|---|
| <BAD-CHOICE>   | which is decimal 1;   |
| <TOO-LONG>     | (for receiver's storage) which is decimal 2;  |
| <WRONG-LENGTH> | (of actual string compared with promised length in <count>) which is decimal 3; or  |
| <OTHER-REASON> | (intended for use only until this document can be updated to include reasons not anticipated by the authors) which is decimal zero (0). |

IAC SB BM <LITERAL> <macro byte> IAC SE

The <macro byte> is to be treated as real data, rather than as representative of the <replacement string>

NOTE: this subcommand cannot be used during Telnet subcommands, since subcommands are defined to end with the next occurrence of "IAC SE". Including this BM subcommand within any Telnet subcommand would therefore prematurely terminate the containing subcommand.

TELNET BYTE MACRO OPTION  
RFC 735, NIC 42083 (Nov. 3, 1977)

<LITERAL> is decimal 4.

IAC SB BM <PLEASE CANCEL> <macro byte> <REASON> IAC SE

The RECEIVER of the defined <macro byte> (i.e., the sender of IAC DO BM) requests the sender of <macro byte> to cancel its definition. <REASON> is the same as for the <REFUSE> subcommand. The <macro byte> sender should (but is not required to) respond by resetting <macro byte> (i.e., sending an IAC SB BM <DEFINE> <macro byte> <1> <macro byte> IAC SE).

If the receiver absolutely insists on cancelling a given macro, the best it can do is to turn off the entire option, with IAC DONT BM, wait for an acknowledging IAC WONT BM and then restart the option, with IAC DO BM. This will reset all other macros as well but it will allow the receiver to REFUSE with code BAD CHOICE if/when the foreign site attempts to redefine the macro in question.

3. Default:

WON'T BM -- DON'T BM

No reinterpretation of data bytes is done.

4. Motivation for the option:

Subcommands for Telnet options currently require a minimum of five characters to be sent over the network (i.e., IAC SB <Option name> IAC SE). For subcommands which are employed infrequently, in absolute numbers and in relation to normal data, this overhead is tolerable. In other cases, however, it is not. For example, data which is sent in a block-oriented fashion may need a "block separator" mark. If blocks are commonly as small as five or ten bytes, then most of the cross-net data will be control information. The BM option is intended as a simple data compression technique, to remove this overhead from the communication channel.

5. Description of the option

The option is enabled through the standard Telnet Option negotiation process. Afterwards, the SENDER of data (the side which sends the IAC WILL BM) is free to define and use mappings between single and replacement NVT characters. Except for the ability to refuse particular definitions, the receiver of data has no control over the definition and use of mappings.

TELNET BYTE MACRO OPTION  
RFC 735, NIC 42083 (Nov. 3, 1977)

The sender (of the WILL BM) is prohibited from using or redefining a <macro byte> until it has received an <ACCEPT> <REFUSE>, or DONT BM, in reply to a <DEFINE>.

NOTE: The Telnet command character IAC (decimal 255) may be a member of a <replacement string> but is the ONLY character which may NOT be defined as a <macro byte>.

Within any Telnet command (i.e., any sequence beginning with IAC) macro replacement may NOT take place. Data are to be interpreted only as their normal character values. This avoids the problem of distinguishing between a character which is to be taken as a <macro byte>, and interpreted as its corresponding <replacement string>, and one which is to be taken as its usual Telnet NVT value. In all other cases, however, <macro byte>s are to be interpreted immediately, as if their corresponding <replacement string>s had actually been sent across the network. Expanded strings are not subject to reinterpretation, so that recursive definitions cannot be made. Telnet commands may be included in <replacement strings>; however, they must be totally contained within the macro or must begin within the macro and terminate outside of it. In particular, they may NOT begin outside a macro and continue or terminate inside one, since no macro replacement takes place while processing any Telnet command.

Note that when skipping data due to Telnet SYNCH (INS/DM) processing, BM macro replacement should still take place, since (for example) "IAC DM" would be a valid <replacement string>.

The <count> in the <DEFINE> subcommand is intended to allow the receiver to allocate storage. IAC interpretation is not over-ridden during BM subcommands so that IAC SE will continue to safely terminate malformed subcommands.

The BM option is notably inefficient with regard to problems during <macro byte> definition and use of <macro byte>s as real data. It is expected that relatively few <macro byte>s will be defined and that they will represent relatively short strings. Since the Telnet data space between decimal 128 and decimal 254 is not normally used, except by implementations employing the original (obsolete) Telnet protocol, it is recommended that <macro byte>s normally be drawn from that pool.



TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

John Day  
RFC 732, NIC 41762 (Sept. 12, 1977)  
Obsoletes RFC 731

TELNET Data Entry Terminal Option

1. Command Name and Code:

DET 20

2. Command Meanings

IAC WILL DET

The sender of this command REQUESTS or AGREES to send and receive subcommands to control the Data Entry Terminal.

IAC WONT DET

The sender of this command REFUSES to send and receive subcommands to control the Data Entry Terminal.

IAC DO DET

The sender of this command REQUESTS or AGREES to send and receive subcommands to control the Data Entry Terminal.

IAC DONT DET

The sender of this command REFUSES to send and receive subcommands to control the Data Entry Terminal.

The DET option uses five classes of subcommands: 1) to establish the requirements and capabilities of the application and the terminal, 2) to format the screen, and to control the 3) edit, 4) erasure, and 5) transmission functions. The subcommands that perform these functions are described below.

The Network Virtual Data Entry Terminal (NVDET)

The NVDET consists of a keyboard and a rectangular display. The keyboard is capable of generating all of the characters of the ASCII character set. In addition, the keyboard may possess a number of function keys which when pressed cause a FN subcommand to be sent. (Although most DET's will support one or more peripheral devices such as a paper tape reader or a printer, this

---

Preceding page blank

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

option does not consider their support. Support of peripheral devices should be treated by a separate option).

The screen of the data entry terminal is a rectangle M characters by N lines. The values of M and N are set by negotiating the Output Line Width and Output Page Size options, respectively. The next writing position (x,y) on the screen (where x is the character position and y is the position of the line on the screen) is indicated by a special display character called the cursor. The cursor may be moved to any position on the screen without disturbing any characters already on the screen. Cursor addressing in existing terminals utilizes several topologies and addressing methods. In order to make the burden of implementation as easy as possible this protocol supports two topologies (the finite plane and the helical torus) and three addressing methods ((x,y); x and y, and relative increments). Since the finite plane with absolute addressing is the least ambiguous and the easiest to translate to and from the others, it is the default scheme used by the NVDET. The torodial form with either relative or absolute addressing is provided for convience.

Also the NVDET provides a mechanism for defining on the screen fields with special attributes. For example, characters entered into these fields may be displayed with brighter intensity, highlighted by reverse video or blinking, or protected from modification by the user. This latter feature is one of the most heavily used for applications where the DET displays a form to be filled out by the user.

The definition of the NVDET uses Telnet option subnegotiations to accomplish all of its functions. Since none of the ASCII characters sent in the data stream have been used to define these functions, the DET option can be used in a "raw" or even "rare" mode. In circumstances where the application program knows what kind of terminal is on the other end, it can send the ASCII characters required to control functions not supported by the option or an implementation. In general keeping all NVDET functions out of the data stream provides better flexibility.

Facility Functions (for detailed semantics see Section 5.)

IAC SB DET <DET facility subcommand><facility map> IAC SE

where <DET facility subcommand> is one 8-bit byte indicating the class of the facilities to be described, and <facility map> is a field of one or two 8-bit bytes containing flags describing the facilities required or desired by the sender.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

The bits of the facility maps are numbered from the right starting at zero. Thus, if bit 2 is set the field will have a decimal value of 4. The values of the field are as follows:

Facility cmd: EDIT FACILITIES    subcommand code: 1

Facility map:    Bit numbers

Toroidal Cursor Addressing	6
Incremental Cursor Addressing	5
Read Cursor Address	4
Line Insert/Delete	3
Char Insert/Delete	2
Back Tab	1
Positive Addressing only	0

where:

If the Toroidal Cursor Addressing bit is set, the sender requests or provides that the SKIP TO LINE and SKIP TO CHAR subcommands be supported.

If the Incremental Cursor Addressing bit is set, the sender requests or provides that the UP, DOWN, LEFT, and RIGHT subcommands be supported.

If the Read Cursor bit is set, the sender requests or provides the READ CURSOR subcommand.

If the Line Insert/Delete bit is set, the sender requests or provides that the LINE INSERT and LINE DELETE subcommands be supported.

If the Char Insert/Delete bit is set, the sender requests or provides that the CHAR INSERT and CHAR DELETE subcommands be supported.

If the Back Tab bit is set, the sender requests or provides that the BACK TAB subcommand be supported.

If the Positive Addressing bit is set, then the sender is informing the receiver that it can only move the cursor in the positive direction. (NOTE: Terminals that have this property also have a Home function to get back to the beginning.)

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

Facility cmd: ERASE FACILITIES subcommand code: 2

Facility map: Bit numbers

Erase Field	4
Erase Line	3
Erase Rest of Screen	2
Erase Rest of Line	1
Erase Rest of Field	0

where:

If a bit of the facility map for this facility command is set, the sender requests or provides the facility indicated by the bit. For a more complete description of each of these functions see the Erase Functions section below.

Facility cmd: TRANSMIT FACILITIES subcommand code: 3

Facility map: Bit numbers

Data Transmit	5
Transmit Line	4
Transmit Field	3
Transmit Rest of Screen	2
Transmit Rest of Line	1
Transmit Rest of Field	0

where:

If a bit of the facility map for this facility command is set, the sender requests or provides the facility indicated by the bit. For a more complete description of each of these functions see the Transmit Functions section below.

Facility cmd: FORMAT FACILITIES subcommand code: 4

facility map: Bit numbers

FN	byte 0	7
Modified		6
Light Pen		5
Repeat		4
Blinking		3
Reverse Video		2
Right Justification		1
Overstrike		0

Protection On/Off	byte 1	6
Protection		5
Alphabetic-only Protection		4
Numeric-only Protection		3
Intensity		0-2

where:

If the FN bit is set, the sender requests or provides the FN subcommand.

If the Modified bit is set, the sender requests or provides the ability to indicate fields that are modified and supports the TRANSMIT MODIFIED subcommand.

If the Light Pen bit is set, the sender requests or provides the support of a light pen, including the Pen Selectable attribute of the DATA FORMAT subcommand.

If the Repeat bit is set the sender requests or provides the REPEAT subcommand.

If the Blinking bit is set, the sender requests or provides the ability to highlight a string of characters by causing them to blink.

If the Reverse Video bit is set, the sender requests or provides the ability to highlight a string of characters by "reversing the video image," i.e., if the characters are normally displayed as black characters on a white background, this is reversed to be white characters on a black background, or vice versa.

If the Right Justification bit is set, the sender requests or provides the ability to cause entries of data to be right justified in the field.

If the Overstrike bit is set, the sender requests or provides the ability to superimpose one character over another on the screen much like a hard copy terminal would do if the print mechanism struck the same position on the paper with different characters.

If the Protection On/Off bit is set, the sender requests or provides the ability to turn on and off field protection.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

If the Protection bit is set, the sender requests or provides the ability to protect certain strings of characters displayed on the screen from being altered by the user of the terminal. Setting this bit also implies that ERASE UNPROTECTED, DATA TRANSMIT, FIELD SEPARATOR, and TRANSMIT UNPROTECTED subcommands (see below) are supported.

If the Alphabetic-only Protection bit is set, the sender requests or provides the ability to constrain the user of the terminal such that he may only enter alphabetic data into certain areas of the screen.

If the Numeric-only Protection bit is set, the sender requests or provides the ability to constrain the user of the terminal such that he may only enter numerical data into certain areas of the screen.

The three bits of the Intensity field will contain a positive binary integer indicating the number of levels of intensity that the sender requests or provides for displaying the data. The value of the 3 bit field should be interpreted in the following way:

- 1 one visible intensity
- 2 two intensities; normal and bright
- 3 three intensities; off, normal, and bright
- >3 >3 intensities; off, and the remaining levels proportioned from dimmest to brightest intensity.

For all of the above commands, if the appropriate bit in <facility map> is not set, then the sender does not request or provide that facility.

#### Editing Functions

IAC SB DET MOVE CURSOR <x><y> IAC SE                            subcommand code: 5

where <x> is an 8-bit byte containing a positive binary integer representing the character position of the cursor, <y> is an 8-bit byte containing a positive binary integer representing the line position of the cursor.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

This subcommand moves the cursor to the absolute screen address (x,y) with the following boundary conditions:

if  $x > M-1$ , set  $x = M-1$  and send an ERROR subcommand

if  $y > N-1$ , set  $y = N-1$  and send an ERROR subcommand

This describes a finite plane topology on the screen.

IAC SB DET SKIP TO LINE <y> IAC SE                            subcommand code: 6

where <y> is a positive 8-bit binary number.

This subcommand moves the cursor to the absolute screen line y. x remains constant. For values of  $y > N-1$

$y = y \bmod N$ .

IAC SB DET SKIP TO CHAR <x> IAC SE                            subcommand code: 7

where <x> is a positive 8-bit binary number.

This subcommand moves the cursor to the absolute character position x. y remains constant, unless  $x > M-1$  in which case:

$x' = (x \bmod M)$

$y' = (y + (x \bmod M) \bmod N)$

where  $x'$  and  $y'$  are the new values of the cursor.

These last two subcommands define a toroidal topology on the screen.

IAC SB DET UP IAC SE    subcommand code: 8

IAC SB DET DOWN IAC SE    subcommand code: 9

IAC SB DET LEFT IAC SE    subcommand code: 10

IAC SB DET RIGHT IAC SE    subcommand code: 11

These subcommands are provided as a convenience for some terminals. The commands UP, DOWN, LEFT, and RIGHT are defined as

UP:         $(x, y) = (x, y - 1 \bmod N)$

DOWN:       $(x, y) = (x, y + 1 \bmod N)$

LEFT:       $(x, y) = (x - 1, y); \text{ if } x = 0 \text{ then } x - 1 = 0$

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

RIGHT:  $(x, y) = (x+1 \bmod M, y)$  and  $y = y+1$  if  $x+1 > M-1$

Note: DOWN, LEFT, and RIGHT cannot always be replaced by the ASCII codes for linefeed, backspace, and space respectively. The latter are format effectors while the former are cursor controls.

IAC SB DET HOME IAC SE    subcommand code: 12

This subcommand positions the cursor to (0,0). This is equivalent to a MOVE CURSOR 0,0 or the sequence SKIP TO LINE 0, SKIP TO CHAR 0. This subcommand is provided for convenience, since most terminals have it as a separate control.

IAC SB DET LINE INSERT IAC SE    subcommand code: 13

This subcommand inserts a line of spaces between lines y (the current line, determined by the position of the cursor) and line y-1. Lines y through N-2 move down one line, i.e. line y becomes line y+1; y+1 becomes y+2, ...; N-2 becomes N-1. Line N-1 is lost off the bottom of the screen. The position of the cursor remains unchanged.

IAC SB DET LINE DELETE IAC SE    subcommand code: 14

This subcommand deletes line y where y is the current line position of the cursor. Lines y+1 through N-1 move up one line, i.e. line y+1 becomes line y; y+2 becomes y+1; ...; N-1 becomes N-2. The N-1st line position is set to all spaces. The cursor position remains unchanged.

IAC SB DET CHAR INSERT IAC SE    subcommand code: 15

This subcommand inserts the next character in the data stream between the xth and x-1st characters, where x is the current character position of the cursor. The xth through M-2nd characters on the line are shifted one character positon to the right. The new character is inserted at the vacated xth position. The M-1st character is lost. The position of the cursor remains unchanged.

IAC SB DET CHAR DELETE IAC SE    subcommand code: 16

This subcommand deletes the character on the screen at the x-th position. The x-th character is removed and the characters x+1 through M-1 are shifted one character position to the left to become the x-th through M-2nd characters. The M-1st character

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

position is left empty. (For most terminals it will be set to a NUL or space.) The cursor position remains unchanged.

IAC SB DET READ CURSOR IAC SE    subcommand code: 17

This subcommand requests the receiver to send the present position of the cursor to the sender.

IAC SB DET CURSOR POSITION <x><y> IAC SE                                    subcommand code: 18

where <x> and <y> are positive 8-bit binary integers.

This subcommand is sent by a Telnet implementation in response to a READ CURSOR subcommand to convey the coordinates of the cursor to the other side. NOTE: x is less than M and y is less than N.

IAC SB DET REVERSE TAB IAC SE    subcommand code: 19

This subcommand causes the cursor to move to the previous tab position. If none exists on the present line, the cursor moves to the previous line and so on until a tab is found or the address (0,0) is encountered. When field protection is in effect the cursor moves to the beginning of the preceding unprotected field.

Transmit Functions (For detailed semantics see Section 5.)

IAC SB DET TRANSMIT SCREEN IAC SE                                    subcommand code: 20

This subcommand causes the terminal to transmit all characters on the screen from position (0,0) to (M-1,N-1). The cursor will be at (0,0) after the operation is complete.

IAC SB DET TRANSMIT UNPROTECTED IAC SE                                    subcommand code: 21

This subcommand causes the terminal to transmit all characters in unprotected fields from position (0,0) to (M-1,N-1). The unprotected fields are separated by the field separator subcommand. The cursor will be at (0,0) or at the beginning of the first unprotected field after the operation is complete.

IAC SB DET TRANSMIT LINE IAC SE    subcommand code: 22

This subcommand causes the terminal to transmit all data on the yth line where y is determined by the present position of the cursor. Data is sent from character position (0,y) to the

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

end-of-line or position (M-1,y) whichever comes first. The cursor position after the transmission is one character position after the end of line condition or the beginning of the next line, (0,y+1).

IAC SB DET TRANSMIT FIELD IAC SE                    subcommand code: 23

This subcommand causes the terminal to transmit all data in the field presently occupied by the cursor. The cursor position after the operation is complete is one character position after the end of the field or, if that position is protected, at the beginning of the next unprotected field.

IAC SB DET TRANSMIT REST OF SCREEN IAC SE                    subcommand code: 24

This subcommand causes the terminal to transmit all characters on the screen from position (x,y) to (M-1,N-1) or until the end of text. (x,y) is the current cursor position. The cursor position after the operation is one character position after the last text character, or (0,0) if the last filled character position is (M-1,N-1).

IAC SB DET TRANSMIT REST OF LINE IAC SE                    subcommand code: 25

This subcommand causes the terminal to transmit all characters on the yth line from position (x,y) to the end of line or (M-1,y) whichever comes first. (x,y) is the current cursor position. The cursor position after the operation is one character position after the last character of the line or the first character of the next line.

IAC SB DET TRANSMIT REST OF FIELD IAC SE                    subcommand code: 26

This subcommand causes the receiver to transmit the rest of the characters in the field currently occupied by the cursor. The cursor position after the operation is at the beginning of the next field.

IAC SB DET TRANSMIT MODIFIED IAC SE                    subcommand code: 27

This subcommand causes the receiver to transmit only those fields which have the modified attribute set. The cursor position after the operation is unchanged.

IAC SB DET DATA TRANSMIT <x><y> IAC SE                    subcommand code: 28

This subcommand is used to preface data sent from the terminal

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

in response to a user action or a TRANSMIT command. The parameters <x> and <y> indicate the initial position of the cursor. See the Transmit Subcommands subsection in Section 5 for more details. A DATA TRANSMIT subcommand may precede an entire transmission with each field being delineated by the FIELD SEPARATOR subcommand, as would be the case in a response to a TRANSMIT UNPROTECTED, or it may precede each field as would be the case in a response to a TRANSMIT MODIFIED.

#### Erase Functions

IAC SB DET ERASE SCREEN IAC SE

subcommand code: 29

This subcommand causes all characters to be removed from the screen. All fields regardless of their attributes are deleted. The cursor position after the operation will be (0,0). Most terminals set the erased characters to either NUL or space characters.

IAC SB DET ERASE LINE IAC SE

subcommand code: 30

This subcommand causes all characters on the yth line to be removed from the screen, where y is the line of the current cursor position. All fields regardless of their attributes are deleted. The cursor position after this operation will be (0,y). NOTE: This operation can be easily simulated by the sequence: LINE DELETE, LINE INSERT. However, the order is important to insure that no data is lost off the bottom of the screen.

IAC SB DET ERASE FIELD IAC SE

subcommand code: 31

This subcommand causes all characters in the field occupied by the cursor to be removed. The cursor position after the operation is at the beginning of the field.

IAC SB DET ERASE REST OF SCREEN IAC SE

subcommand code: 32

This subcommand causes all characters from position (x,y) to (M-1,N-1) to be removed from the screen. All fields regardless of their attributes are deleted. The cursor position after the operation is unchanged. This is equivalent to doing an ERASE REST OF LINE plus a LINE DELETE for lines greater than y.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

IAC SB DET ERASE REST OF LINE IAC SE                            subcommand code: 33

This subcommand causes all characters from position (x,y) to (M-1,y) to be removed from the screen. All fields regardless of their attributes are deleted. The cursor position after the operation is unchanged.

IAC SB DET ERASE REST OF FIELD IAC SE                            subcommand code: 34

This subcommand causes all characters from position (x,y) to the end of the current field to be removed from the screen. The cursor position after the operation is unchanged.

IAC SB DET ERASE UNPROTECTED IAC SE                            subcommand code: 35

This subcommand causes all characters on the screen in unprotected fields to be removed from the screen. The cursor position after the operation is at (0,0) or, if that position is protected, at the beginning of the first unprotected field.

#### Format Functions

IAC SB DET FORMAT DATA <format map><count> IAC SE  
    subcommand code: 36

where <format map> is a two-byte field containing the following flags:

Byte 0	
Blinking	7
Reverse Video	6
Right Justification	5
Protection	3-4
Intensity	0-2
Byte 1	
Modified	1
Pen Selectable	0

where:

If the Blinking bit is set, the following field of <count> characters should have the Blinking attribute applied to it by the receiver.

If the Reverse Video bit is set, the following field of <count> characters should be displayed by the receiver with video reversed.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

If the Right Justification bit is set, the input entered into the field of <count> characters should be right justified.

The Protection field is two bits wide and may take on the following values:

0	no protection
1	protected
2	alphabetic only
3	numeric only

The protection attribute specifies that the other side may modify any character (no protection), modify no characters (protected), enter only alphabetical characters (A-Z, and a-z) (alphabetic only), or enter only numerical characters (0-9, +, ., and -) (numeric only) in the following field of <count> bytes.

The Intensity field is 3 bits wide and should be interpreted in the following way:

The values 0-6 should be used as an indication of the relative brightness to be used when displaying the characters in or entered into the following field <count> characters wide. The number of levels of brightness available should have been obtained previously by the Format Facility subcommand. The exact algorithm for mapping these values to the available levels of intensity is left to the implementors. A value of 7 in the intensity field indicates that the brightness should be off, and any characters in or entered into the field should not be displayed.

If the Modified bit is set, the field is considered to have been modified and will be transmitted in response to a TRANSMIT MODIFIED subcommand.

If the Pen Selectable bit is set, the field can be selected with the light pen. NOTE: Use of the light pen should be the subject of another Telnet option.

<count> is 2 bytes that should be interpreted as a positive 16-bit binary integer representing the number of characters following this command which are affected by it.

Data sent to the terminal or the Using Host for unwritten areas of the screen not in the scope of the count should be displayed with the default values of the format map. The default values

are No Blinking, Normal Video, No Justification, No Protection and Normal Intensity. For example, suppose a FORMAT DATA subcommand was sent to the terminal with attributes Blinking and Protected and a count of 5 followed by the string "Name: John Doe". The string "Name:" would be protected and blinking, but the string "John Doe" would not be.

This subcommand is used to format data to be displayed on the screen of the terminal. The <format map> describes the attributes that the field <count> bytes wide should have. This field is to start at the position of the cursor when the command is acted upon. The next <count> displayable characters in the data stream are used to fill the field. Subsequent REPEAT subcommands may be used to specify the contents of this field. If the sender specifies attributes that have not been agreed upon by the use of the Format Facility subcommand, the Telnet process should send an Error Subcommand to the sender, but format the screen as if the bit had not been set.

IAC SB DET REPEAT <count><char> IAC SE                    subcommand code: 37

where <count> is a positive 8-bit binary integer. <char> is an 8-bit byte containing an ASCII character.

This subcommand is used to perform data compression on data being transferred to the terminal by encoding strings of identical characters as the character and a count. The repeated characters may be part of a field specified.

IAC SB DET SUPPRESS PROTECTION <negotiation> IAC SE                    subcommand code: 38

where <negotiation> may have the values of the Telnet option negotiation:

251	WILL
252	WONT
253	DO
254	DONT

This subcommand is used to suppress the field protection in a non-destructive manner. Many data entry terminals provide the means by which protection may be turned on and off without modifying the contents of the screen or the terminal's memory. Thus, the protection may be turned off and back on without retransmitting the form.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

The default setting of the option is that protection is on, in other words

```
IAC SB DET SUPPRESS PROTECTION WONT IAC SE
IAC SB DET SUPPRESS PROTECTION DONT IAC SE
```

Negotiation of this subcommand follows the same rules as negotiations of the Telnet options.

IAC SB DET FIELD SEPARATOR IAC SE                    subcommand code: 39

It is necessary when transmitting only the unprotected portion of the screen to provide a means for delimiting the fields. Existing DETs use a variety of ASCII characters such as Tab, Group Separator, Unit Separator, etc. In order to maintain transparency of the NVDET this subcommand is used to separate the fields. Clearly, this incurs rather high overhead. This overhead can be avoided by using the Byte Macro Option (see Appendix 3).

#### Miscellaneous Commands

IAC SB DET FN <code> IAC SE                    subcommand code: 40

where: <code> is one byte.

Many data-entry terminals provide a set of "function" keys which when pressed send a one-character command to the server. This subcommand describes such a facility. The values of the <code> field are defined by the user and server. The option merely provides the means to transfer the information.

IAC SB DET ERROR <cmd> <error code> IAC SE                    subcommand code: 41

where:

<cmd> is a byte containing the subcommand code of the subcommand in error.

<error code> is a byte containing an error code.

(For a list of the defined error codes see Appendix 2.)

This subcommand is provided to allow DET option implementations to report errors they detect to the corresponding Telnet process. At this point it is worth reiterating that the philosophy of this option is that when an error is detected it

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

should be reported; however, the implementation should attempt its best effort to carry out the intent of the subcommand or data in error.

### 3. Default and Minimal Implementation Specifications

#### Default

WON'T DET -- DON'T DET

Neither host wishes to use the Data Entry Terminal option.

#### Minimal Implementation

DET EDIT FACILITIES  
DET ERASE FACILITIES  
DET TRANSMIT FACILITIES  
DET FORMAT FACILITIES  
DET MOVE CURSOR <x><y>  
DET HOME  
DET ERASE SCREEN  
DET TRANSMIT SCREEN  
DET FORMAT DATA  
DET ERROR <cmd> <error code>

In the case of formatting the data, the minimal implementation should be able to support a low and high level of intensity and protection for all or no characters in a field. These functions, however, are not required.

The minimal implementation also requires that the Output Line Width and Output Page Size Telnet options be supported.

#### 4. Motivation

The Telnet protocol was originally designed to provide a means for scroll-mode terminals, such as the standard teletype, to communicate with processes through the network. This was suitable for the vast majority of terminals and users at that time. However, as use of the network has increased into other areas, especially areas where the network is considered to provide a production environment for other work, the desires and requirements of the user community have changed. Therefore, it is necessary to consider supporting facilities that were not initially supported. This Telnet option attempts to do that for applications that require data entry terminals.

This option in effect defines the Network Virtual Data Entry Terminal. Although the description of this option is quite long, this does not imply that the Telnet protocol is a poor vehicle for this facility. Data Entry Terminals are rather complex and varied in their abilities. This option attempts to support both the minimal set of useful functions that are either common to all or can be easily simulated and the more sophisticated functions supplied in some terminals.

Unlike most real data entry terminals where the terminal functions are encoded into one or more characters of the native character set, this option performs all such controls within the Telnet subnegotiation mechanism. This allows programs that are intimately familiar with the kind of terminal they are communicating with to send commands that may not be supported by either the option or the implementation. In other words, it is possible to operate in a "raw" or at least "rare" mode using as much of the option as necessary.

Although many data entry terminals support a variety of peripheral devices such as printers, cassettes, etc. it is beyond the scope of this option to entertain such considerations. A separate option should be defined to handle this aspect of these devices.

## 5. Description

### General Notes

All implementations of this option are required to support a certain minimal set of the subcommands for this option. Section 3 contains a complete list of the subcommands in this minimal set. In keeping with the Telnet protocol philosophy that an implementation should not have to be able to parse commands it does not implement, every subcommand of this option is either in the minimal set or is covered by one of the facility subcommands. An implementation must "negotiate" with its correspondent for permission to use subcommands not in the minimal set before using them. For details of this negotiation process see the section below on facility subcommands.

Most data entry terminals are used in a half duplex mode. (Although most DET's on the market can be used either as data entry terminals or as standard interactive terminals, we are only concerned here with their use as DET's.) When this option is used, it is suggested that the following Telnet options be refused: Echo, Remote Controlled Transmission and Echoing, and Suppress Go-Ahead. However, this option could be used to support a simple full duplex CRT based application using the basic cursor control functions provided here. For these cases, one or more of the above list of options might be required. (Support of sophisticated interactive calligraphic applications is beyond the scope of this option and should be done by another option or the Network Graphics Protocol.)

In RFC 728, it was noted that a synch sequence can cause undesired interactions between Telnet Control functions and the data stream. A synch sequence causes data, but not control functions, to be flushed. If a control function which has an effect on the data immediately following it is present in the data stream when a synch sequence occurs, the control function will have its effect, not on the intended data, but on the data immediately following the Data Mark. These DET subcommands are susceptible to this pitfall:

```
CHAR INSERT
DATA TRANSMIT
FORMAT DATA
```

The undesired interactions are best avoided by the receiver of the synch sequence deleting these subcommands and all data associated with them before continuing to process the control functions.

This implies that the Data Mark should not occur in the middle of the data associated with these subcommands.

#### Facility Subcommands

These four subcommands are used by the User and Server implementations to negotiate the subcommands and attributes of the terminal that may be utilized. This negotiation can be viewed as the terminal (User Host) indicating what facilities are provided and the Server Host (or application program) indicating what facilities are desired.

When Sent: A Server Telnet implementation using the DET option must send a facility subcommand requesting the use of a particular subcommand or terminal attribute not in the minimal implementation before the first use of that subcommand or attribute. The User Telnet implementation should respond as quickly as possible with its reply. Neither the User nor Server are required to negotiate one subcommand at a time. Also, a Telnet implementation responding to a facility subcommand is not required to give permission only for that subcommand. It may send a format map indicating all facilities of that class which it supports. However, a Telnet implementation requesting facilities must send a facility subcommand before its first use of the subcommand regardless of whether earlier negotiations have indicated the facility is provided. The facility cannot be used until a corresponding facility subcommand has been received. There are no other constraints on when the facility subcommands may be sent. In particular, it is not necessary for an application to know at the beginning of a session all facilities that it will use.

Action When Received: There are two possible actions that may be taken when a facility subcommand is received depending on whether the receiver is a requestor or a provider (User).

Requestor: When a facility subcommand is received by a requestor and it is in the state of Waiting for a Reply, it should go into the state of Not Waiting. It should then take the facility map it had sent and form the logical intersection with the facility map received. (For the Intensity attribute, one should take the minimum of the number received and the number requested.) The result indicates the facilities successfully negotiated. NOTE: if the receiver is not in the Waiting for Reply state, then this is the provider case described next.

Provider: When a facility subcommand is received, it should send a facility subcommand with a facility map of the facilities it

provides as soon as possible. It should then determine what new facilities it is providing for the Requestor by forming the logical intersection of the facility map received and the one sent.

NOTE: Although in most cases the requestor will be the Server Host and the provider will be the User Host supporting the terminal, this distinction may not always be true.

#### Transmit Subcommands

There are two kinds of transmit subcommands: those used to request that data be sent to the requestor, and one to preface data sent to the requestor. The first kind allow the requestor to control when, from where and to some degree how much data is transmitted from the terminal. Their explanation is straightforward and may be found in Section 2.

Data may be sent from the terminal as a result of two events: the user of the terminal caused the transmission or in response to a transmit subcommand. Some programs may wish to know from where on the screen the transmission began. (This is reasonable, since the terminal user may move the cursor around considerably before transmitting.) Other programs may not need such information. The DATA TRANSMIT subcommand is provided in case this function is needed. When used this subcommand prefaces data coming from the terminal. The parameters  $\langle x \rangle$  and  $\langle y \rangle$  give the screen coordinates of the beginning of the transmission.  $\langle x \rangle$  must be less than or equal to M-1 and  $\langle y \rangle$  must be less than or equal to N-1. It is assumed that all data between this DATA TRANSMIT and the next one starts at the coordinates given by the first subcommand and continues filling each line thereafter according to the constraints of the screen and the format effectors in the data. Thus an intelligent or sloppy user-host DET implementation (depending on your point of view) need only include a DATA TRANSMIT subcommand when the new starting point is different from the last ending point.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

## 6. Sample Interaction

The nomenclature of RFC 726 will be used to describe this example.  
To quote that RFC:

"S:" is sent from serving host to using host.  
"U:" is sent from using host to serving host.  
"T:" is entered by the terminal user.  
"P:" is printed on the terminal.

Text surrounded by square brackets ([]) is commentary. Text surrounded by angle brackets (<>) is to be taken as a single unit, e.g., carriage return is <cr>, and the decimal value 27 is represented <27>.

We assume that the user has established the Telnet connection, logged on, and an application program has just been started either by the user directly or through a canned start up procedure. The presentation on the page is meant to merely group entities together and does not imply the position of message boundaries. One should assume that any part of the dialogue may be sent as one or many messages. The first action of the program or Telnet is to negotiate the DET option:

S: <IAC><DO><DET>

U: <IAC><WILL><DET>

S:<IAC><DO><OUTPUT PAGE SIZE>

[First negotiate the screen size. In this case we are asking the user the size of the terminal. This could have been done before the DET option was negotiated.]

U:<IAC><WILL><NAOP>

U:<IAC><SB><NAOP><DR><25><IAC><SE>

S:<IAC><SB><NAOP><DS><0><IAC><SE>

S:<IAC><DO><OUTPUT LINE WIDTH>

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

U:<IAC><SB><NAOL><DR><80><IAC><SE> [Defines the screen to be 25 lines by 80 characters. The server may use this information when formatting the screen.]

S:<IAC><SB><NAOL><DS><0><IAC><SE>

S:<IAC><SB><DET><FORMAT FACILITIES><Repeat><Protection, 3 Levels Intensity><IAC><SE> [Now set the terminal attributes.]

U:<IAC><SB><DET><FORMAT FACILITIES><Repeat, Blinking><Protection, 3 Levels Intensity><IAC><SE>

S:<IAC><SB><DET><ERASE SCREEN><IAC><SE> [Erase the screen and start sending the form.]

<IAC><SB><DET><FORMAT DATA><Protection=1, Intensity=1><0><5><IAC><SE>Name:

<IAC><SB><DET><MOVE CURSOR><0><1><IAC><SE>

<IAC><SB><DET><FORMAT DATA><Protection=1, Intensity=1><0><8><IAC><SE>Address:

<IAC><SB><MOVE CURSOR><0><4><IAC><SE>

<IAC><SB><DET><FORMAT DATA><Protection=1, Intensity=1><0><17><IAC><SE>Telephone number:

<IAC><SB><DET><MOVE CURSOR><32><4><IAC><SE>

<IAC><SB><DET><FORMAT DATA><Protection=1, Intensity=1><0><24><IAC><SE>Social Security Number:

<IAC><SB><DET><FORMAT DATA><Protection=1, Intensity=7><0><11><IAC><SE> [Establish a field that doesn't display what is typed into it.]

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

```
<IAC><SB><DET><MOVE CURSOR><32><5><IAC><SE>
<IAC><SB><DET><FORMAT FACILITIES>
<Blinking><0><IAC><SE>                                [Get permission to use
                                                               Blinking Attribute.]
```

U:<IAC><SB><DET><FORMAT FACILITIES>
<Repeat, Blinking><Protection,
3 Levels Intensity><IAC><SE>

S:<IAC><SB><DET><FORMAT DATA>
<Blinking=1, Protection=1,
Intensity=1><0><29><IAC><SE>

Your SSN will not be printed.

```
<IAC><SB><DET><HOME><IAC><SE>
<IAC><GA>
```

The previous exchange has placed a form on the screen that looks like:

Name:	
Address:	
Telephone Number:	Social Security Number: "Your SSN will not be printed."

where the quoted string is blinking.

The terminal user is now free to fill in the form provided. He positions the cursor at the beginning of the first field (this usually is done by hitting the tab key) and begins typing. We do not show this interaction since it does not generate any interaction with the User Telnet program or the network. After the terminal user has completed filling in the form, he strikes the transmit key to send the unprotected part of the form, but first the User Telnet program negotiates the Byte Macro Option to condense the Field Separator subcommand:

```
U:<IAC><DO><BM>                                [Negotiate Byte
                                                               MacroOption.]
```

S:<IAC><WILL><BM> [Define decimal 166 to be
 the Field Separator
 subcommand (see Appendix
 3)]

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

U:<IAC><SB><BM><DEFINE>  
<166><6><IAC SB DET FIELD  
SEPARATOR IAC SE><IAC><SE>

S:<IAC><SB><BM><ACCEPT><166><IAC><SE> [The server accepts the  
macro.]

U:<IAC><SB><DET><DATA TRANSMIT><0><6><IAC><SE>  
John Doe <166> 1515 Elm St., Urbana, Il 61801  
<166> 217-333-9999 <166> 123-45-6789 <166>

S:<IAC><SB><DET><ERASE SCREEN><IAC><SE>  
Thank you.

And so on.

Appendix 1 - Subcommands, opcodes and syntax

1	EDIT FACILITIES	<Facility map>
2	ERASE FACILITIES	<Facility map>
3	TRANSMIT FACILITIES	<Facility map>
4	FORMAT FACILITIES	<Facility map 1> <Facility map 2>
5	MOVE CURSOR	<x> <y>
6	SKIP TO LINE	<y>
7	SKIP TO CHAR	<x>
8	UP	
9	DOWN	
10	LEFT	
11	RIGHT	
12	HOME	
13	LINE INSERT	
14	LINE DELETE	
15	CHAR INSERT	
16	CHAR DELETE	
17	READ CURSOR	
18	CURSOR POSITION	<x><y>
19	REVERSE TAB	
20	TRANSMIT SCREEN	
21	TRANSMIT UNPROTECTED	
22	TRANSMIT LINE	
23	TRANSMIT FIELD	
24	TRANSMIT REST OF SCREEN	
25	TRANSMIT REST OF LINE	
26	TRANSMIT REST OF FIELD	
27	TRANSMIT MODIFIED	
28	DATA TRANSMIT <x><y>	
29	ERASE SCREEN	
30	ERASE LINE	
31	ERASE FIELD	
32	ERASE REST OF SCREEN	
33	ERASE REST OF LINE	
34	ERASE REST OF FIELD	
35	ERASE UNPROTECTED	
36	FORMAT DATA <format map>	
37	REPEAT <count><char>	
38	SUPPRESS PROTECTION <negotiation>	
39	FIELD SEPARATOR	
40	FN <code>	
41	ERROR <cmd><error code>	

Appendix 2 - Error Codes

- 1 Facility not previously negotiated.
- 2 Illegal subcommand code.
- 3 Cursor Address Out of Bounds.
- 4 Undefined FN value.
- 5 Can't negotiate acceptable line width.
- 6 Can't negotiate acceptable page length.
- 7 Illegal parameter in subcommand.
- 8 Syntax error in parsing subcommand.
- 9 Too many parameters in subcommand.
- 10 Too few parameters in subcommand.
- 11 Undefined parameter value
- 12 Unsupported combination of Format Attributes

### Appendix 3 - Use of the Byte Macro Option

One of the major drawbacks of the DET option is that because the functions are encoded as Telnet option subnegotiations a fairly high overhead is incurred. A function like Character Insert which is encoded as a single byte in most terminals requires six bytes in the DET option. Originally the only other solution that would have accomplished the same transparency that the use of subcommands provides would have been to define additional Telnet control functions. However, since this would entail modification of the Telnet protocol itself, it was felt that this was not a wise solution. Since then the Telnet Byte Macro Option (RFC 729) has been defined. This option allows the user and server Telnets to map an arbitrary character string into a single byte which is then transferred over the net. Thus the Byte Macro Option provides the means for implementations to avoid the overhead for heavily used subcommands. The rest of this appendix suggests how the Byte Macro Option should be applied to the DET option.

In keeping with the specification of the Byte Macro Option, macro bytes will be chosen from the range 128 to 239. For the DET option, it is suggested that macro bytes be chosen by adding the subcommand code to 128. In addition, an unofficial DET subcommand might be defined indicating that each side was willing to support macro bytes for all subcommands (but not necessarily support all of the subcommands themselves) according to this algorithm. This subcommand would be:

IAC SB DET DET-MACRO <negotiation> IAC SE      subcommand code: 25<sup>4</sup>

where <negotiation> may have the values of the Telnet option negotiation:

251	WILL
252	WONT
253	DO
254	DONT

This subcommand is sent by a Telnet implementation to indicate its willingness to adopt byte macros for all of the DET subcommands according to the following algorithm:

The macro byte for subcommand i will be i+128 and will represent the following string for parameterless subcommands:

IAC SB DET <subcommand code> IAC SE

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

and the following string for subcommands with parameters:

IAC SB DET <subcommand code>

The default setting for this subcommand is that the macros are not in effect, in other words,

IAC SB DET DET-MACRO WONT IAC SE  
IAC SB DET DET-MACRO DONT IAC SE

Negotiation of this subcommand follows the same rules as negotiations of the Telnet options.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

References

1. ADM-1 Interactive Display Terminal Operator's Handbook  
Lear-Siegler, Inc. 7410-31.
2. ADM-Interactive Display Terminal Operator's Handbook  
Lear-Siegler, Inc. EID, 1974.
3. Burroughs TD 700/800 Reference Manual, Burroughs Corp., 1973
4. Burroughs TD 820 Reference Manual, Burroughs Corp. 1975.
5. CC-40 Communications Station: General Information Manual.  
Computer Communication, Inc. Pub. No. MI-1100. 1974.
6. Crocker, David. "Telnet Byte Macro Option," RFC 729, 1977.
7. Data Entry Virtual Terminal Protocol for Euronet, DRAFT, 1977.
8. Day, John. "A Minor Pitfall in the Telnet Protocol," RFC 728,  
1977.
9. Hazeltine 2000 Desk Top Display Operating Instructions.  
Hazeltine IB-1866A, 1870.
10. How to Use the Consul 980: A Terminal Operator's Guide and  
Interface Manual. Applied Digital Data Systems, Inc. 98-3000.
11. How to Use the Consul 520: A Terminal Operator's Guide and  
Interface Manual. Applied Digital Data Systems, Inc. 52-3000.
12. Honeywell 7700 Series Visual Information Projection (VIP)  
Systems: Preliminary Edition. 1973.
13. An Introduction to the IBM 3270 Information Display System. IBM  
GA27-2739-4. 1973.
14. Naffah, N. "Protocole Appareil Virtuel type Ecran" Reseau  
Cyclades. TER 536. 1976.
15. Postel, Jon and Crocker, David. "Remote Controlled Transmission  
and Echoing Telnet Option", RFC 726 NIC 39237, Mar. 1977.
16. Schicker, Peter. "Virtual Terminal Protocol (Proposal 2). INWG  
Protocol Note #32., 1976.

TELNET Data Entry Terminal Option  
RFC 732, NIC 41762 (Sept. 13, 1977)

17. UNISCOPE Display Terminal : Programmer Reference . Sperry-Univac UP-7807 Rev. 2, 1975.
18. Universal Terminal System 400: System Description. Sperry-Univac UP-8357, 1976.
19. Walden, David C. "Telnet Output Line Width Option." NIC # 20196, 1973, also in ARPANET Protocol Handbook, 1976.
20. Walden, David C. "Telnet Output Page Size" NIC # 20197, 1973, also in ARPANET Protocol Handbook, 1976.

TELNET SUPDUP Option  
RFC 736, NIC 42213 (Oct. 31, 1977)

Mark Crispin (SU-AI)  
RFC 736, NIC 42213 (Oct. 31, 1977)

TELNET SUPDUP Option

1. Command name and code.

SUPDUP 21

2. Command meanings.

IAC WILL SUPDUP

The sender of this command REQUESTS permission to, or confirms that it will, use the SUPDUP display protocol

IAC WON'T SUPDUP

The sender of this command REFUSES to use the SUPDUP protocol.

IAC DO SUPDUP

The sender of this command REQUESTS that the receiver use, or grants the receiver permission to use, the SUPDUP protocol.

IAC DON'T

The sender of this command DEMANDS that the receiver not use the SUPDUP protocol.

3. Default.

WON'T SUPDUP

DON'T SUPDUP

i.e., the SUPDUP display protocol is not in use.

TELNET SUPDUP Option  
RFC 736, NIC 42213 (Oct. 31, 1977)

4. Motivation for the option.

Since the publication of RFC 734, I have been requested to design an option to the TELNET protocol to provide for SUPDUP service. This option allows a host to provide SUPDUP service on the normal TELNET socket (27 octal) instead of 137 (octal) which is the normal SUPDUP ICP socket.

5. Description of the option.

A user TELNET program which wishes to use the SUPDUP display protocol instead of the NVT terminal service should send an IAC DO SUPDUP. If the server is willing to use the SUPDUP display protocol, it should respond with IAC WILL SUPDUP; otherwise it should refuse with IAC WONT SUPDUP.

For hosts which normally provide SUPDUP terminal services, the server can send IAC WILL SUPDUP upon ICP which the user may then accept or refuse.

If the SUPDUP option is in effect, no further TELNET negotiations are allowed. They are meaningless, since SUPDUP has its own facilities to perform the functions that are needed. Hence, octal 377 will become an ordinary transmitted character (in this case an invalid %TD code) instead of an IAC.

Following the mutual acceptance of the SUPDUP option, the SUPDUP negotiation proceeds as described in RFC 734.

SUPDUP Protocol  
RFC 734, NIC 41953 (Oct. 7, 1977)

Mark Crispin (SU-AI)  
RFC 734, NIC 41953 (Oct. 7, 1977)

SUPDUP Protocol

INTRODUCTION

This document describes the SUPDUP protocol, a highly efficient display Telnet protocol. It originally started as a private protocol between the ITS systems at MIT to allow a user at any one of these systems to use one of the others as a display. At the current writing, SUPDUP user programs also exist for Data Disc and Datamedia displays at SU-AI and for Datamedias at SRI-KL. The author is not aware of any SUPDUP servers other than at the four MIT ITS sites.

The advantage of the SUPDUP protocol over an individual terminal's protocol is that SUPDUP defines a "virtual" or "software" display terminal that implements relevant cursor motion operations. The protocol is not built on any particular display terminal but rather on the set of functions common to all display terminals; hence it is completely device-independent. In addition, the protocol also provides for terminals which cannot handle certain operations, such as line or character insert/delete. In fact, it is more than this. It provides for terminals which are missing any set of features, all the way down to model 33 Teletypes.

The advantage over the TELNET protocol is that SUPDUP takes advantage of the full capabilities of display terminals, although it also has the ability to run printing terminals.

It is to be noted that SUPDUP operates independently from TELNET; it is not an option to the TELNET protocol. In addition, certain assumptions are made about the server and the user programs and their capabilities. Specifically, it is assumed that the operating system on a server host provides all the display-oriented features of ITS. However, a server may elect not to do certain display operations available in SUPDUP; the SUPDUP protocol is far-reaching enough so that the protocol allows terminals to be handled as well as that host can handle terminals in general. Of course, if a host does not support display terminals in any special way, there is no point in bothering to implement a SUPDUP server since TELNET will work just as well.

A more complete description of the display facilities of SUPDUP and ITS can be found by FTPing the online file .INFO.;ITS TTY from ARPANET host MIT-AI (host 206 octal, 134 decimal). For more

SUPDUP Protocol  
RFC 734, NIC 41953 (Oct. 7, 1977)

information, the mailing address for SUPDUP is "(BUG SUPDUP) at MIT-AI". If your mail system won't allow you to use parentheses, use Bug-SUPDUP@MIT-AI.

#### BACKGROUND

The SUPDUP protocol originated as the internal protocol used between parts of ITS, and between ITS and "intelligent" terminals. Over the network, a user host acts like an intelligent terminal programmed for ITS.

The way terminal output works in ITS is as follows: The user program tells the system to do various operations, such as printing characters, clearing the screen, moving the cursor, etc. These operations are formed into 8-bit characters (using the %TD codes described below) and stored into a buffer. At interrupt level, as the terminal demands output, characters are removed from the buffer and translated into terminal dependent codes. At this time padding and cursor motion optimization are also done.

In some cases, the interrupt side does not run on the same machine as the user program. SUPDUP terminals have their "interrupt side" running in the user host. When SUPDUP is run between two ITS's, the SUPDUP user and server programs and the network simply move characters from the buffer in the server machine to the buffer in the user machine. The interrupt side then runs on the user machine just as if the characters had been generated locally.

Due to the highly interactive characteristics of both the SUPDUP protocol and the ITS system, all transactions are strictly character at a time and all echoing is remote. In addition, all padding and cursor control optimization must be done by the user.

Because this is also the internals of ITS, the right to change it any time if necessary to provide new features is reserved by MIT. In particular, the initial negotiation is probably going to be changed to transmit additional variables, and additional %TD codes may be added at any time. User programs should ignore those they don't know about.

The following conventions are used in this document: function keys (ie, keys which represent a "function" rather than a "graphic character") are in upper case in square brackets. Prefix keys (ie, keys which generate no character but rather are held down while typing another character to modify that character) are in upper case in angle brackets. Hence "<CONTROL><META>[LINE FEED]" refers to the character generated when both the CONTROL and META keys are held down

while a LINE FEED is typed. Case should be noted; <CONTROL>A refers to a different character from <CONTROL>a. Finally, all numbers which do not explicitly specify a base (ie, octal or decimal) should be read as octal unless the number is immediately followed by a period, in which case it is decimal.

#### INITIALIZATION

The SUPDUP server listens on socket 137 octal. ICP proceeds in the normal way for establishing 8-bit connections. After the ICP is completed, the user side sends several parameters to the server side in the form of 36.-bit words. Each word is sent through the 8-bit connection as six 6-bit bytes, most-significant first. Each byte is in the low-order 6 bits of a character. The first word is the negative of the number of variables to follow in the high order 18. bits (the low-order 18. bits are ignored), followed by the values of the TCTYP, TTYOPT, TCMXV, TCMXH, and TTYROL terminal descriptor variables (these are the names they are known by at ITS sites). These variables are 36.-bit binary numbers and define the terminal characteristics for the virtual terminal at the REMOTE host.

The count is for future compatibility. If more variables need to be sent in the future, the server should assume "reasonable" default values if the user does not specify them. PDP-10 fans will recognize the format of the count (ie, -count,,0) as being an AOBJN pointer. At the present writing there are five variables hence this word should be -5,,0.

The TCTYP variable defines the terminal type. It MUST be 7 (%TNSFW). Any other value is a violation of protocol.

The TTYOPT variable specifies what capabilities or options the user's terminal has. A bit being true implies that the terminal has this option. This variable also includes user options which the user may wish to alter at his or her own discretion; these options are included since they may be specified along with the terminal capabilities in the initial negotiation. See below for the relevant TTYOPT bits.

The TCMXV variable specifies the screen height in number of lines.

The TCMXH variable specifies the line width in number of characters. This value is one less than the screen width (ITS indicates line overflow by outputting an exclamation point at the end of the display line before moving to the next line). NOTE: the terminal must not do an automatic CRLF when a character is printed in the rightmost

SUPDUP Protocol

RFC 734, NIC 41953 (Oct. 7, 1977)

column. If this is unavoidable, the user SUPDUP must decrement the width it sends by one.

NOTE: Setting either the TCMXV or TCMXH dimension greater than 128 will work, but will have some problems as coordinates are sometimes represented in only 7 bits. The main problems occur in the SUPDUP protocol when sending the cursor position after an output reset and in ITS user programs using the display position codes ^PH and ^PV.

The TTYROL variable specifies the "glitch count" when scrolling. This is the number of lines to scroll up when scrolling is required. If zero, the terminal is not capable of scrolling. 1 is the usual value, but some terminals glitch up by more than one line when they scroll.

Following the transmission of the terminal options by the user, the server should respond with an ASCII greeting message, terminated with a %TDNOP code (%TD codes are described below). All transmissions from the server after the %TDNOP are either printing characters or virtual terminal display codes.

The user and the server now both communicate using the intelligent terminal protocol (described below) from the user and %TD codes from the server. The user has two commands in addition to these; they are escaped by sending 300 (octal). If following the escape is a 301 (octal), the server should attempt to log off the remote job (generally this is sent immediately before the user disconnects, so this logout procedure should be done regardless of the continuing integrity of the connection). If the character following the escape is a 302 (octal), all ASCII characters following up to a null (000 octal) are interpreted as "console location" which the server can handle as it pleases. No carriage return or line feed should be in the console location text. Normally this is saved away to be displayed by the "who" command when other users ask where this user is located.

#### TTYOPT FUNCTION BITS

The relevant TTYOPT bits for SUPDUP usage follow. The values are given in octal, with the left and right 18-bit halves separated by ",,," as in the usual PDP-10 convention.

Bit name	Value	Meaning
%TOALT	200000,,0	characters 175 and 176 are converted to altmode (033) on input.
%TOERS	40000,,0	this terminal is capable of selectively erasing its screen. That is, it supports the %TDEOL, the %TDDLF, and (optionally) the %TDEOF operations. For terminals which can only do single-character erasing, see %TOOVR.
%TOMVB	10000,,0	this terminal is capable of backspacing (ie, moving the cursor backwards).
%TOSAI	4000,,0	this terminal has the Stanford/ITS extended ASCII graphics character set.
%TOOVR	1000,,0	this terminal is capable of overprinting; if two characters are displayed in the same position, they will both be visible, rather than one replacing the other.
		Lack of this capability but the capability to backspace (see %TOMVB) implies that the terminal can do single character erasing by overstriking with a space. This allows terminals without the %TOERS capability to have display-style "rubout processing", as this capability depends upon either %TOERS or [%TOMVB and not %TOOVR].
%TOMVU	400,,0	this terminal is capable of moving the cursor upwards.
%TOLWR	20,,0	this terminal's keyboard is capable of generating lowercase characters; this bit is mostly provided for programs which want to know this information.

SUPDUP Protocol  
RFC 734, NIC 41953 (Oct. 7, 1977)

Bit name	Value	Meaning
%TOFCI	10,,0	this terminal's keyboard is capable of generating CONTROL and META characters as described below.
%TOLID	2,,0	this terminal is capable of doing line insert/delete operations, ie, it supports %TDILP and %TDDLP.
%TOCID	1,,0	this terminal is capable of doing character insert/delete operations, ie, it supports %TDICP and %TDDCP.
%TPCBS	0,,40	this terminal is using the "intelligent terminal protocol". THIS BIT MUST BE ON.
%TPORS	0,,10	the server should process output resets instead of ignoring them. IT IS HIGHLY RECOMMENDED THAT THIS BIT BE ON; OTHERWISE THERE MAY BE LARGE DELAYS IN ABORTING OUTPUT.

The following bits are user option bits. They may be set or not set at the user's discretion. The bits that are labelled "normally on" are those that are normally set on when a terminal is initialized (ie, by typing [CALL] on a local terminal).

Bit name	Value	Meaning
%TOCLC	100000,,0	convert lower-case input to upper case. Many terminals have a "shift lock" key which makes this option useless. NORMALLY OFF.
%TOSA1	2000,,0	characters 001-037 should be displayed using the Stanford/ITS extended ASCII graphics character set instead of uparrow followed by 100+character. NORMALLY OFF.
%TOMOR	200,,0	the system should provide "%%MORE%%" processing when the cursor reaches the bottom line of the screen. %%MORE%% processing is described in ITS TTY. NORMALLY ON.
%TOROL	100,,0	the terminal should scroll when attempting output below the bottom line of the screen instead of wrapping around to the top. NORMALLY OFF.

## INPUT -- THE INTELLIGENT TERMINAL PROTOCOL

NOTE: only the parts of the intelligent terminal protocol relevant to SUPDUP are discussed here. For more information, read ITS TTY.

### CHARACTER SETS

There are two character sets available for use with SUPDUP; the 7-bit character set of standard ASCII, and the 12-bit character set of extended ASCII. Extended ASCII has 5 high order or "bucky" bits on input and has graphics for octal 000-037 and 177 (see the section entitled "Stanford/ITS character set" for more details). The two character sets are identical on output since the protocol specifies that the host should never send the standard ASCII formatting characters (ie, TAB, LF, VT, FF, CR) as formatting characters; the characters whose octal values are the same as these formatting characters are never output unless the user job has these characters enabled (setting %TOSAI and %TOSA1 generally does this).

Input differs dramatically between the 7-bit and 12-bit character sets. In the 7-bit character set, all characters input whose value is 037 octal or less are assumed to be (ASCII) control characters. In the 12-bit character set, there are 5 "bucky" bits which may be attached to the character. The two most important of these are CONTROL and META, which form a 9-bit character set. TOP is used to distinguish between printing graphics in the extended character set and ASCII controls. The other two are reserved and should be ignored. Since both 7-bit and 12-bit terminals are commonly in use, 0001, 0301, and 0341 are considered to be <CONTROL>A on input by most programs, while 4001 is considered to be downwards arrow.

### MAPPING BETWEEN CHARACTER SETS

Many programs and hosts do not process 12-bit input. In this case, 12-bit input is folded down to 7-bit as follows: TOP and META are discarded. If CONTROL is on, then if the 7-bit part of the character specifies a lower case alphabetic it is converted to upper case; then if the 7-bit part is between 077 and 137 the 100 bit is complemented or if the 7-bit part is 040 the 040 bit is subtracted (that's right, <CONTROL>? is converted to [RUBOUT] and <CONTROL>[SPACE] is converted to [NULL]). In any case the CONTROL bit is discarded, and the remainder is treated as a 7-bit ASCII character. It should be noted that in this case downwards arrow is read by the program as standard ASCII <CONTROL>A.

Servers which expect 12-bit input and are told to use the 7-bit character set should do appropriate unfolding from the 7-bit character set to 12-bit. It is up to the individual server to decide upon the unfolding scheme. On ITS, user programs that use the 12-bit character set generally have an alternative method for 7-bit; this often takes the form of prefix characters indicating that the next character should be "controllified" or "metized", etc.

#### BUCKY BITS

Under normal circumstances, characters input from the keyboard are sent to the foreign host as is. There are two exceptions; the first occurs when an octal 034 character is to be sent; it must be quoted by being sent twice, because 034 is used as an escape character for protocol commands. The second exception occurs when %TOFCI is set and a character with non-zero bucky bits is to be sent. In this case, the character, which is in the 12-bit form:

Name	Value	Description
%TXTOP	4000	This character has the [TOP] key depressed.
%TXSFL	2000	Reserved, must be zero.
%TXSFT	1000	Reserved, must be zero.
%TXMTA	400	This character has the [META] key depressed.
%TXCTL	200	This character has the [CONTROL] key depressed.
%TXASC	177	The ASCII portion of the character

is sent as three bytes. The first byte is always 034 octal (that is why 034 must be quoted). The next byte contains the "bucky bits", ie, the %TXTOP through %TXCTL bits, shifted over 7 bits (ie, %TXTOP becomes 20) with the 100 bit on. The third byte contains the %TXASC part of the character. Hence the character <CONTROL><META>[LINE FEED] is sent as 034 103 012.

#### OUTPUT RESETS

The intelligent terminal protocol also is involved when a network interrupt (INR/INS) is received by the user program. The user program should increment a count of received network interrupts when this happens. It should not do any output, and if possible abort any

output in progress, if this count is greater than zero (NOTE: the program MUST allow for the count to go less than zero).

Since the server no longer knows where the cursor is, it suspends all output until the user informs it of the cursor position. This also gives the server an idea of how much was thrown out in case it has to have some of the aborted output displayed at a later time. The user program does this when it receives a %TDORS from the server. When this happens it should decrement the "number of received network interrupts" count described in the previous paragraph and then send 034 followed by 020, the vertical position, and the horizontal position of where the cursor currently is located on the user's screen.

#### OUTPUT -- DISPLAY PROTOCOL (%TD CODES)

Display output is somewhat simpler. Codes less than 200 octal are printing characters and are displayed on the terminal (see the section describing the "Stanford/ITS character set"). Codes greater than or equal to 200 (octal) are known as "%TD codes", so called since their names begin with %TD. The %TD codes that are relevant to SUPDUP operation are listed here. Any other code received should be ignored, although a bug report might be sent to the server's maintainers. Note that the normal ASCII formatting characters (011 - 015) do NOT have their formatting sense under SUPDUP and should not occur at all unless the Stanford/ITS extended ASCII character set is in use (i.e., %TOSAI is set in the TTYOPT word).

For cursor positioning operations, the top left corner is (0,0), i.e., vertical position 0, horizontal position 0.

%TD code	Value	Meaning
%TDMOV	200	General cursor position code. Followed by four bytes; the first two are the "old" vertical and horizontal positions and may be ignored. The next two are the new vertical and horizontal positions. The cursor should be moved to this position.
		On printing consoles (non %TOMVU), the old vertical position may differ from the true vertical position; this can occur when scrolling. In this case, the user program should set its idea of the old vertical position to what the %TDMOV says and then proceed. Hence a %TDMOV with an old vpos of

20. and a new vpos of 22. should always move the "cursor" down two lines. This is used to prevent the vertical position from becoming infinite.

%TDMV1	201	An internal cursor motion code which should not be seen; but if it is, it has two argument bytes after it and should be treated the same as %TDMV0.
%TDEOF	202	Erase to end of screen. This is an optional function since many terminals do not support this. If the terminal does not support this function, it should be treated the same as %TDEOL.
		%TDEOF does an erase to end of line, then erases all lines lower on the screen than the cursor. The cursor does not move.
%TDEOL	203	Erase to end of line. This erases the character position the cursor is at and all positions to the right on the same line. The cursor does not move.
%TDDLF	204	Clear the character position the cursor is on. The cursor does not move.
%TDCRL	207	If the cursor is not on the bottom line of the screen, move cursor to the beginning of the next line and clear that line. If the cursor is at the bottom line, scroll up.
%TDNOP	210	No-op; should be ignored.
%TDORS	214	Output reset. This code serves as a data mark for aborting output much as IAC DM does in the ordinary TELNET protocol.
%TDQOT	215	Quotes the following character. This is used when sending 8-bit codes which are not %TD codes, for instance when loading programs into an intelligent terminal. The following character should be passed through intact to the terminal.

SUPDUP Protocol  
RFC 734, NIC 41953 (Oct. 7, 1977)

%TDFS	216	Non-destructive forward space. The cursor moves right one position; this code will not be sent at the end of a line.
%TDMVO	217	General cursor position code. Followed by two bytes; the new vertical and horizontal positions.
%TDCLR	220	Erase the screen. Home the cursor to the top left hand corner of the screen.
%TDBEL	221	Generate an audio tone, bell, whatever.
%TDILP	223	Insert blank lines at the cursor; followed by a byte containing a count of the number of blank lines to insert. The cursor is unmoved. The line the cursor is on and all lines below it move down; lines moved off the bottom of the screen are lost.
%TDDLP	224	Delete lines at the cursor; followed by a count. The cursor is unmoved. The first line deleted is the one the cursor is on. Lines below those deleted move up. Newly-created lines at the bottom of the screen are blank.
%TDICP	225	Insert blank character positions at the cursor; followed by a count. The cursor is unmoved. The character the cursor is on and all characters to the right on the current line move to the right; characters moved off the end of the line are lost.
%TDDCP	226	Delete characters at the cursor; followed by a count. The cursor is unmoved. The first character deleted is the one the cursor is on. Newly-created characters at the end of the line are blank.
%TDBOW	227	Display black characters on white screen. HIGHLY OPTIONAL.
%TDRST	230	Reset %TDBOW and such any future options.

#### STANFORD/ITS CHARACTER SET

This section describes the extended ASCII character set. It originated with the character set developed at SAIL but was modified for 1968 ASCII.

This character set only applies to terminals with the %TOSAI and %TOFCI bits set in its TTYOPT word. For non-%TOSAI terminals, the standard ASCII printing characters are the only available output characters. For non-%TOFCI terminals, the standard ASCII characters are the only available input characters.

#### PRINTING CHARACTERS

The first table describes the printing characters. For output, the 7-bit code is sent (terminal operations are performed by %TD codes). For input, the characters with values 000-037 and 177 must have the %TXTOP bit on to indicate the graphic is intended rather than a function or ASCII control.

##### Value Character

4000	centered dot
4001	downward arrow
4002	alpha
4003	beta
4004	logical AND
4005	logical NOT
4006	epsilon
4007	pi
4010	lambda
4011	gamma
4012	delta
4013	uparrow
4014	plus-minus
4015	circle-plus
4016	infinity
4017	partial delta
4020	proper subset (left horseshoe)
4021	proper superset (right horseshoe)
4022	intersection (up horseshoe)
4023	union (downward horseshoe)
4024	universal quantifier
4025	existential quantifier
4026	circle-x
4027	double arrow
4030	left arrow

Value	Character
4031	right arrow
4032	not-equal
4033	lozenge (diamond)
4034	less-than-or-equal
4035	greater-than-or-equal
4036	equivalence
4037	logical OR
0040	first standard ASCII character (space)
..	.. .
0176	last standard ASCII character (tilde)
4177	integral

#### FUNCTION KEYS AND SPECIAL CHARACTERS

In addition, the following special characters exist for input only. These characters are function keys rather than printing characters; however, some of these characters have some format effect or graphic which they echo as; the host, not the SUPDUP program, handles any such mappings.

Value	Character	Usual echo	Usual Function
0000	[NULL]		
0010	[BACK SPACE]		text formatting
0011	[TAB]		text formatting
0012	[LINE FEED]		text formatting
0013	[VT]		text formatting
0014	[FORM]		text formatting
0015	[RETURN]		text formatting
0032	[CALL]	uparrow-Z	escape to system
0033	[ALTMODE]	lozenge or \$	special activation
0037	[BACK NEXT]	uparrow-underscore	monitor command prefix
0177	[RUBOUT]		character delete
4101	[ESCAPE]		local terminal command
4102	[BREAK]		local subsystem escape
4103	[CLEAR]		
4110	[HELP]		requests a help message

#### BUCKY BITS

For all input characters, the following "bucky bits" may be added to the character. Their interpretation depends entirely upon the host. <TOP> is not listed here, as it has been considered part of the character in the previous tables.

SUPDUP Protocol  
RFC 734, NIC 41953 (Oct. 7, 1977)

<CONTROL> is different from ASCII CTRL, however, many programs may request the operating system to map such characters to the ASCII forms (with the <TOP> bit off). In this case <META> is ignored.

Value Key

2000	Reserved
1000	Reserved
0400	<META>
0200	<CONTROL>

ACKNOWLEDGEMENTS

Richard M. Stallman (RMS@MIT-AI) and David A. Moon (Moon@MIT-MC) of the MIT-AI and MIT-MC systems staff wrote the source documentation and the wonderful ITS terminal support that made this protocol possible. It must be emphasized that this is a functional protocol which has been in operation for some years now.

In addition, Moon, Stallman, and Michael McMahon (MMcM@SRI-KL) provided many helpful comments and corrections to this document.

For further reference, the sources for the known currently existing SUPDUP user programs are available online as:

[MIT-AI] SYSENG;SUPDUP > for the ITS monitor,  
[SU-AI] SUPDUP.MID[NET,MRC] for the SAIL monitor,  
[SRI-KL] <MMcM>SD.FAI for the TOPS-20 monitor.

The source for the known currently existing SUPDUP server program is:

[MIT-AI] SYSENG;TELSER > for the ITS monitor.

These programs are written in the MIDAS and FAIL dialects of PDP-10 assembly language.

TELNET Extended-Options - List option  
NIC 16239

TELNET Extended-Options - List option

1. Command name and code.

EXTENDED-OPTIONS-LIST (EXOPL) 255

2. Command meanings.

IAC DO EXOPL

The sender of this command REQUESTS that the receiver of this command begin negotiating, or confirms that the receiver of this command is expected to begin negotiating, TELNET options which are on the "Extended Options List."

IAC WILL EXOPL

The sender of this command requests permission to begin negotiating, or confirms that it will begin negotiating, TELNET options which are on the "Extended Options List."

IAC WON'T EXOPL

The sender of this command REFUSES to negotiate, or to continue negotiating, options on the "Extended Options List."

IAC DON'T EXOPL

The sender of this command DEMANDS that the receiver conduct no further negotiation of options on the "Extended Options List."

IAC SB EXOPL <subcommand>

The subcommand contains information required for the negotiation of an option of the "Extended Options List." The format of the subcommand is discussed in section 5 below.

3. Default

WON'T EXOPL, DON'T EXOPL

i.e., negotiation of options on the "Extended Options List" is not permitted.

TELNET Extended-Options - List option  
NIC 16239

4. Motivation for the option

Eventually, a 257th TELNET option will be needed and there is currently no way to support it. The proposed option will extend the option list for another 256 options in a manner which is easy to implement. The option is proposed now, rather than later (probably much later), in order to reserve the option number (255).

5. An abstract description of the option.

The EXOPL option has five subcommand codes: WILL, WON'T, DO, DON'T, and SB. They have exactly the same meanings as the TELNET commands with the same names, and are used in exactly the same way. For consistency, these subcommand codes will have the same values as the TELNET command codes (250-254). Thus, the format for negotiating a specific option on the "Extended Options List" (once both parties have agreed to use it) is:

IAC SB EXOPL DO/DON'T/WILL/WON'T/<option code>

Once both sides have agreed to use the specific option specified by <option code>, subnegotiation may be required. In this case the format to be used is:

IAC SB EXOPL SB <option code> <parameters>

**USAS**  
**X3.4-1968**  
Revision of  
**X3.4-1967**

# **USA Standard Code for Information Interchange**

Sponsor  
**Business Equipment Manufacturers Association**

Approved October 10, 1968  
**United States of America Standards Institute**



# Contents

SECTION	PAGE
1. Scope .....	6
2. Standard Code .....	6
3. Character Representation and Code Identification .....	7
4. Legend .....	7
4.1 Control Characters.....	7
4.2 Graphic Characters .....	7
5. Definitions .....	8
5.1 General .....	8
5.2 Control Characters.....	8
5.3 Graphic Characters .....	9
6. General Considerations .....	9
Appendixes	
Appendix A Design Considerations for the Coded Character Set .....	11
A1. Introduction .....	11
A2. Considerations Affecting the Standard .....	11
A3. Set Size .....	11
A4. Set Structure .....	11
A5. Choice of Graphics .....	11
A6. Graphic Subset Structure.....	12
A7. Control Subset Content and Structure .....	13
Appendix B Notes on Application .....	13
B1. Introduction .....	13
B2. Character Substitutions .....	13
B3. Related Larger and Smaller Sets .....	14
B4. International Considerations .....	14
B5. Communications Considerations .....	14
Appendix C Original Criteria .....	14
C1. Introduction .....	14
C2. Criteria .....	15
Appendix D Terminology.....	15

Preceding page blank

# USA Standard Code for Information Interchange

## 1. Scope

This coded character set is to be used for the general interchange of information among information processing systems, communication systems, and associated equipment.

## 2. Standard Code

b <sub>7</sub>				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1	
b <sub>6</sub>				b <sub>5</sub>								
b <sub>4</sub> b <sub>3</sub> b <sub>2</sub> b <sub>1</sub>					COLUMN							
↓	↓	↓	↓	ROW ↓								
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	\$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	:
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	—	o	DEL

### 3. Character Representation and Code Identification

The standard 7-bit character representation, with  $b_7$  the high-order bit and  $b_1$  the low-order bit, is shown below:

EXAMPLE: The bit representation for the character "K," positioned in column 4, row 11, is

$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$
1	0	0	1	0	1	1

The code table position for the character "K" may also be represented by the notation "column 4, row 11" or alternatively as "4/11." The decimal equivalent of the binary number formed by bits  $b_7$ ,  $b_6$ , and  $b_5$ , collectively, forms the column number, and the decimal equivalent of the binary number formed by bits  $b_4$ ,  $b_3$ ,  $b_2$ , and  $b_1$ , collectively, forms the row number.

The standard code may be identified by the use of the notation ASCII or USASCII.

The notation ASCII (pronounced as'-key) or USASCII (pronounced you-sas'-key) should ordinarily be taken to mean the code prescribed by the latest issue of the standard. To explicitly designate a particular (perhaps prior) issue, the last two digits of the year of issue may be appended, as, "ASCII 63" or "USASCII 63".

### 4. Legend

#### 4.1 Control Characters

NUL	Null	DLE	Data Link Escape (CC)
SOH	Start of Heading (CC)	DC1	Device Control 1
STX	Start of Text (CC)	DC2	Device Control 2
ETX	End of Text (CC)	DC3	Device Control 3
EOT	End of Transmission (CC)	DC4	Device Control 4 (Stop)
ENQ	Enquiry (CC)	NAK	Negative Acknowledge (CC)
ACK	Acknowledge (CC)	SYN	Synchronous Idle (CC)
BEL	Bell (audible or attention signal)	ETB	End of Transmission Block (CC)
BS	Backspace (FE)	CAN	Cancel
HT	Horizontal Tabulation (punched card skip) (FE)	EM	End of Medium
LF	Line Feed (FE)	SUB	Substitute
VT	Vertical Tabulation (FE)	ESC	Escape
FF	Form Feed (FE)	FS	File Separator (IS)
CR	Carriage Return (FE)	GS	Group Separator (IS)
SO	Shift Out	RS	Record Separator (IS)
SI	Shift In	US	Unit Separator (IS)
		DEL	Delete <sup>1</sup>

---

NOTE: (CC) Communication Control  
(FE) Format Effector  
(IS) Information Separator

<sup>1</sup>In the strict sense, DEL is not a control character. (See 5.2.)

## 4.2 Graphic Characters

<u>Column/Row</u>	<u>Symbol</u>	<u>Name</u>
2/0	SP	Space (Normally Non-Printing)
2/1	!	Exclamation Point
2/2	"	Quotation Marks (Diaeresis <sup>2</sup> )
2/3	#	Number Sign <sup>3,4</sup>
2/4	\$	Dollar Sign
2/5	%	Percent
2/6	&	Ampersand
2/7	'	Apostrophe (Closing Single Quotation Mark; Acute Accent <sup>2</sup> )
2/8	(	Opening Parenthesis
2/9	)	Closing Parenthesis
2/10	*	Asterisk
2/11	+	Plus
2/12	,	Comma (Cedilla <sup>2</sup> )
2/13	-	Hyphen (Minus)
2/14	.	Period (Decimal Point)
2/15	/	Slant
3/10	:	Colon
3/11	;	Semicolon
3/12	<	Less Than
3/13	=	Equals
3/14	>	Greater Than
3/15	?	Question Mark
4/0	@	Commercial At <sup>3</sup>
5/11	[	Opening Bracket <sup>3</sup>
5/12	\	Reverse Slant <sup>3</sup>
5/13	]	Closing Bracket <sup>3</sup>
5/14	^	Circumflex <sup>2,3</sup>
5/15	—	Underline
6/0	~	Grave Accent <sup>2,3</sup> (Opening Single Quotation Mark)
7/11	{	Opening Brace <sup>3</sup>
7/12		Vertical Line <sup>3</sup>
7/13	}	Closing Brace <sup>3</sup>
7/14	~	Overline <sup>3</sup> (Tilde <sup>2</sup> ; General Accent <sup>2</sup> )

<sup>2</sup>The use of the symbols in 2/2, 2/7, 2/12, 5/14, 6/0, and 7/14 as diacritical marks is described in Appendix A, A5.2.

<sup>3</sup>These characters should not be used in international interchange without determining that there is agreement between sender and recipient. (See Appendix B4.)

<sup>4</sup>In applications where there is no requirement for the symbol #, the symbol £ may be used in position 2/3.

## 5. Definitions

### 5.1 General

**(CC) Communication Control:** A functional character intended to control or facilitate transmission of information over communication networks.

**(FE) Format Effector:** A functional character which controls the layout or positioning of information in printing or display devices.

**(IS) Information Separator:** A character which is used to separate and qualify information in a logical sense. There is a group of four such characters, which are to be used in a hierarchical order.

### 5.2 Control Characters

**NUL (Null):** The all-zeros character which may serve to accomplish time fill and media fill.

**SOH (Start of Heading):** A communication control character used at the beginning of a sequence of characters which constitute a machine-sensible address or routing information. Such a sequence is referred to as the "heading." An STX character has the effect of terminating a heading.

**STX (Start of Text):** A communication control character which precedes a sequence of characters that is to be treated as an entity and entirely transmitted through to the ultimate destination. Such a sequence is referred to as "text." STX may be used to terminate a sequence of characters started by SOH.

**ETX (End of Text):** A communication control character used to terminate a sequence of characters started with STX and transmitted as an entity.

**EOT (End of Transmission):** A communication control character used to indicate the conclusion of a transmission, which may have contained one or more texts and any associated headings.

**ENQ (Enquiry):** A communication control character used in data communication systems as a request for a response from a remote station. It may be used as a "Who Are You" (WRU) to obtain identification, or may be used to obtain station status, or both.

**ACK (Acknowledge):** A communication control character transmitted by a receiver as an affirmative response to a sender.

**BEL (Bell):** A character for use when there is a need to call for human attention. It may control alarm or attention devices.

**BS (Backspace):** A format effector which controls the movement of the printing position one printing space backward on the same printing line. (Applicable also to display devices.)

**HT (Horizontal Tabulation):** A format effector which controls the movement of the printing position to the next in a series of predetermined positions along the printing line. (Applicable also to display devices and the skip function on punched cards.)

**LF (Line Feed):** A format effector which controls the movement of the printing position to the next printing line. (Applicable also to display devices.) Where appropriate, this character may have the meaning "New Line" (NL), a format effector which controls the movement of the printing point to the first printing position on the next printing line. Use of this convention requires agreement between sender and recipient of data.

**VT (Vertical Tabulation):** A format effector which controls the movement of the printing position to the next in a series of predetermined printing lines. (Applicable also to display devices.)

**FF (Form Feed):** A format effector which controls the movement of the printing position to the first predetermined printing line on the next form or page. (Applicable also to display devices.)

**CR (Carriage Return):** A format effector which controls the movement of the printing position to the first printing position on the same printing line. (Applicable also to display devices.)

**SO (Shift Out):** A control character indicating that the code combinations which follow shall be interpreted as outside of the character set of the standard code table until a Shift In character is reached.

**SI (Shift In):** A control character indicating that the code combinations which follow shall be interpreted according to the standard code table.

**DLE (Data Link Escape):** A communication control character which will change the meaning of a limited number of contiguously following characters. It is used exclusively to provide supplementary controls in data communication networks.

**DC1, DC2, DC3, DC4 (Device Controls):** Characters for the control of ancillary devices associated with data processing or telecommunication systems, more especially switching devices "on" or "off." (If a single "stop" control is required to interrupt or turn off ancillary devices, DC4 is the preferred assignment.)

**NAK (Negative Acknowledge):** A communication control character transmitted by a receiver as a negative response to the sender.

**SYN (Synchronous Idle):** A communication control character used by a synchronous transmission system in the absence of any other character to provide a signal from which synchronism may be achieved or retained.

**ETB (End of Transmission Block):** A communication control character used to indicate the end of a block of data for communication purposes. ETB is used for blocking data where the block structure is not necessarily related to the processing format.

**CAN (Cancel):** A control character used to indicate that the data with which it is sent is in error or is to be disregarded.

**EM (End of Medium):** A control character associated with the sent data which may be used to identify the physical end of the medium, or the end of the used, or wanted, portion of information recorded on a medium.

(The position of this character does not necessarily correspond to the physical end of the medium.)

**SUB (Substitute):** A character that may be substituted for a character which is determined to be invalid or in error.

**ESC (Escape):** A control character intended to provide code extension (supplementary characters) in general information interchange. The Escape character itself is a prefix affecting the interpretation of a limited number of contiguously following characters.

**FS (File Separator), GS (Group Separator), RS (Record Separator), and US (Unit Separator):** These information separators may be used within data in optional fashion, except that their hierarchical relationship shall be: FS is the most inclusive, then GS, then RS, and US is least inclusive. (The content and length of a File, Group, Record, or Unit are not specified.)

**DEL (Delete):** This character is used primarily to "erase" or "obliterate" erroneous or unwanted characters in perforated tape. (In the strict sense, DEL is not a control character.)

### 5.3 Graphic Characters

**SP (Space):** A normally non-printing graphic character used to separate words. It is also a format effector which controls the movement of the printing position, one printing position forward. (Applicable also to display devices.)

## 6. General Considerations

**6.1** This standard does not define the means by which the coded set is to be recorded in any physical medium, nor does it include any redundancy or define techniques for error control. Further, this standard does not define data communication character structure, data communication formats, code extension techniques, or graphic representation of control characters.

**6.2** Deviations from the standard may create serious difficulties in information interchange and should be used only with full cognizance of the parties involved.

**6.3** The relative sequence of any two characters, when used as a basis for collation, is defined by their binary values.

**6.4** No specific meaning is prescribed for any of the graphics in the code table except that which is understood by the users. Furthermore, this standard does not specify a type style for the printing or display of the various graphic characters. In specific applications, it may be desirable to employ distinctive styling of individual graphics to facilitate their use for specific purposes as, for example, to stylize the graphics in code positions 2/1 and 5/14 into those frequently associated with logical OR (1) and logical NOT ( $\neg$ ), respectively.

**6.5** The appendixes to this standard contain additional information on the design and use of this code.

# Appendices

(These Appendices are not a part of USA Standard Code for Information Interchange, X3.4-1968, but are included to facilitate its use.)

## Appendix A Design Considerations for the Coded Character Set

### A1. Introduction

The standard coded character set is intended for the interchange of information among information processing systems, communication systems, and associated equipment.

### A2. Considerations Affecting the Code

There were many considerations that determined the set size, set structure, character selection, and character placement of the code. Among these were (not listed in order of priority):

- (1) Need for adequate number of graphic symbols
- (2) Need for adequate number of device controls, format effectors, communication controls, and information separators
- (3) Desire for a non-ambiguous code, i.e., one in which every code combination has a unique interpretation
- (4) Physical requirements of media and facilities
- (5) Error control considerations
- (6) Special interpretation of the all-zeros and all-ones characters
- (7) Ease in the identification of classes of characters
- (8) Data manipulation requirements
- (9) Collating conventions
  - (a) Logical
  - (b) Historical
- (10) Keyboard conventions
  - (a) Logical
  - (b) Historical
- (11) Other set sizes
- (12) International considerations
- (13) Programming languages
- (14) Existing coded character sets.

### A3. Set Size

A 7-bit set is the minimum size that will meet the requirements for graphics and controls in applications involving general information interchange.

### A4. Set Structure

**A4.1** In discussing the set structure it is convenient to divide the set into 8 columns and 16 rows as indicated in this standard.

**A4.2** It was considered essential to have a dense subset which contained only graphics. For ease of identification this graphic subset was placed in six contiguous columns.

**A4.3** The first two columns were chosen for the controls for the following three reasons:

- (1) The character NUL by its definition has the location 0/0 in the code table. NUL is broadly considered a control character.
- (2) The representations in column 7 were felt to be most susceptible to simulation by a particular class of transmission error—one which occurs during an idle condition on asynchronous systems.
- (3) To permit the considerations of graphic subset structure described in A6 to be satisfied, the two columns of controls had to be adjacent.

**A4.4** The character set was structured to enable the easy identification of classes of graphics and controls.

### A5. Choice of Graphics

**A5.1** Included in the set are the numerals 0 through 9, upper and lower cases of the alphabetic letters A through Z, and those punctuation, mathematical, and business symbols considered most useful. The set includes a number of characters commonly encountered in programming languages. In particular, all the COBOL and FORTRAN graphics are included.

**A5.2** In order to permit the representation of languages other than English, one diacritical (or accent) mark has been included, and provision has been made for the use of five punctuation symbols alternately as diacritical marks. The pairing of these punctuation symbols with their corresponding diacritical marks was done to facilitate the design of a typeface which would be acceptable for both uses.

These arrangements are:

Col/Row	Code Table Symbol	Punctuation	<u>Use</u>	Diacritical
2/2	"	Quotation Marks		Diæresis
2/7	'	Apostrophe		Acute Accent
2/12	,	Comma		Cedilla
5/14	^	(None)		Circumflex
6/0	\	Opening Single		
		Quotation Mark	Grave Accent	
7/14	-	Overline	Tilde*	

\*May also be used for other accents not specifically provided.

**A5.3** The character *overline* is shown as it is in the code table to suggest a means of avoiding confusion with *underline*, and to reflect its use as *tilde*. The character *vertical line* is shown as it is in the code table to avoid confusion with the solid vertical bar frequently used as a logical operator, which may be found in some systems as a graphic stylization of *exclamation point*.

## A6. Graphic Subset Structure

**A6.1** The basic structure of the dense graphic subset was influenced by logical collating considerations, the requirements of simply related 6-bit sets, and the needs of typewriter-like devices. For information processing, it is desirable that the characters be arranged in such a way as to minimize both the operating time and the hardware components required for ordering and sequencing operations. This requires that the relative order of characters, within classes, be such that a simple comparison of the binary codes will result in information being ordered in a desired sequence.

**A6.2** Conventional usage requires that SP (*space*) be ahead of any other symbol in a collatable set. This permits a name such as "JOHNS" to collate ahead of a name such as "JOHNSON." The requirement that punctuation symbols such as *comma* also collate ahead of the alphabet ("JOHNS, A" should also collate before "JOHNSON") establishes the special symbol locations, including SP, in the first column of the graphic subset.

**A6.3** To simplify the design of typewriter-like devices, it is desirable that there be only a common 1-bit difference between characters to be paired on keytops. This, together with the requirements for a contiguous alphabet, and the collating requirements outlined above, resulted in the placement of the alphabet in the last four columns of the graphic subset and the placement of the numerals in the second column of the graphic subset.

**A6.4** It is expected that devices having the capability of printing only 64 graphic symbols will continue to be important. It may be desirable to arrange these devices to print one symbol for the bit pattern of both upper and lower case of a given alphabetic letter. To facilitate this, there should be a single bit difference between the upper and lower case representations of any given letter. Combined with the requirement that a given case of the alphabet be contiguous, this dictated the assignment of the alphabet, as shown, in columns 4 through 7.

**A6.5** To minimize ambiguity caused by the use of a 64-graphic device as described above, it is desirable, to the degree possible, that each character in column 6 or 7 differ little in significance from the corresponding character in column 4 or 5. In certain cases, this was not possible.

**A6.6** The assignment of *reverse slant* and *vertical line*, the *brackets* and *braces*, and *circumflex* and *overline* were made with a view to the considerations of A6.5.

**A6.7** The resultant structure of "specials" (S), "digits" (D), and "alphabetics" (A) does not conform to the most prevalent collating convention (S-A-D) because of other more demanding code requirements.

**A6.8** The need for a simple transformation from the set sequence to the prevalent collating convention was recognized, and dictated the placement of some of the "specials" within the set. Specifically, those special symbols, viz., *ampersand* (&), *asterisk* (\*), *comma* (,), *hyphen* (-), *period* (.), and *slant* (/), which are most often used as identifiers for ordering information and which normally collate ahead of both the alphabet and the numerals, were not placed in the column containing the numbers, so that the entire numeric column could be rotated via a relatively simple transformation to a position higher than the alphabet. The sequence of the aforementioned "specials" was also established to the extent practical to conform to the prevalent collating convention.

**A6.9** The need for a useful 4-bit numeric subset also played a role in the placement of characters. Such a 4-bit subset, including the digits and the symbols *asterisk*, *plus* (+), *comma*, *hyphen*, *period*, and *slant*, can easily be derived from the code.

**A6.10** Considerations of other domestic code sets, including the Department of Defense former standard 8-bit data transmission code ("Fieldata"-1961), as well as international requirements, played an important role in deliberations that resulted in the code. The selection and grouping of the symbols *dollar sign* (\$), *percent* (%), *ampersand* (&), *apostrophe* ('), *less than* (<), *equals* (=), and *greater than* (>) facilitate contraction to either a business or scientific 6-bit subset. The position of these symbols, and of the symbols *comma*, *hyphen*, *period*, and *slant*, facilitates achievement of commonly accepted pairing on a keyboard. The historic pairing of *question mark* and *slant* is preserved and the *less than* and *greater than* symbols, which have comparatively low usage, are paired with *period* and *comma* so that in dual-case keyboard devices where it is desired to have *period* and *comma* in both cases, the *less than* and *greater than* symbols are the ones displaced. Provision was made for the accommodation of alphabets containing more than 26 letters and for 6-bit contraction

by the location of low-usage characters in the area following the alphabet. In addition, the requirement for the digits 10 and 11 used in sterling monetary areas was considered in the placement of the *asterisk*, *plus*, *semicolon*, and *colon*, so that the 10 and 11 could be substituted for the *semicolon* and *colon*.

## A7. Control Subset Content and Structure

**A7.1** The control characters included in the set are those required for the control of terminal devices, input and output devices, format, or communication transmission and switching on a general enough basis to justify inclusion in a standard set.

**A7.2** Many control characters may be considered to fall into the following categories:

- (1) Communication Controls
- (2) Format Effectors
- (3) Device Controls
- (4) Information Separators.

To the extent practical controls of each category were grouped in the code table. The structure chosen also facilitates the contraction of the set to a logically related 6-bit set.

**A7.3** The information separators (FS, GS, RS, US) identify boundaries of various elements of information, but differ from punctuation in that they are primarily intended to be machine sensible. They were arranged in accordance with an expected hierarchical use, and the lower end of the hierarchy is contiguous in binary order with SP (space) which is sometimes used as a machine-sensible separator. Subject to this hierarchy the exact nature of their use within data is not specified.

**A7.4** The character SYN (Synchronous Idle) was located so that its binary pattern in serial transmission was unambiguous as to character framing, and also to optimize certain other aspects of its communication usage.

**A7.5** ACK (Acknowledge) and NAK (Negative Acknowledge) were located so as to gain the maximum practical protection against mutation of one into the other by transmission errors.

**A7.6** The function "New Line" (NL) was associated with LF (rather than with CR or with a separate character) to provide the most useful combinations of functions through the use of only two character positions, and to allow the use of a common end-of-line format for both printers having separate CR-LF functions and those having a combined (i.e., NL) function. This sequence would be CR-LF, producing the same result on printers of both classes, and would be useful during conversion of a system from one method of operation to the other.

## Appendix B Notes on Application

### B1. Introduction

**B1.1** The standard code was developed to provide for information interchange among information processing systems, communications systems, and associated equipment. In a system consisting of equipment with several local or native codes, maximum flexibility will be achieved if each of the native codes is translated to the standard whenever information interchange is desired.

**B1.2** Within any particular equipment, system, or community of activity, it may be necessary to substitute characters. For example, some systems may require special graphic symbols and some devices may require special control codes. (Design efforts on the standard code included consideration of these types of adaptations.) So-called "secular sets" produced by such substitutions, although not conforming to this standard, may nonetheless be consonant with it if substitutions are made in accordance with the guidelines of B2.

### B2. Character Substitutions

**B2.1** Any character substitution will result in a coded character set which does not conform to this standard.

**B2.2** It is recommended that when a character is substituted in the code table for a standard character, the standard character should not be reassigned elsewhere in the table. Such a substitute character, once assigned, should not be subsequently reassigned elsewhere.

**B2.3** It is recommended that graphic substitutions be made only in the graphic area and control substitutions only in the control area. Any substitution involving a control should be made only with full cognizance of all possible operational effects.

**B2.4** It should be noted that this standard specifies, for each position of the code table, the information represented by the character and not necessarily the precise action taken by the recipient when the character is received. In the case of graphics, considerable variation in the actual shape printed or displayed may be appropriate to different units, systems, or fields of application. In the case of controls, the action performed is dependent upon the use for which the particular system is intended, the application to which it is being put, and a number of conventions established by the user or designer—some system-wide and some unique to a particular unit.

**B2.5** Typical examples of diversity in execution not necessarily contrary to this standard are:

(1) A number of graphic symbols, other than that used in the code table, are used for *ampersand* in various type styles; still other symbols may be more appropriate to electronic display devices. The use of such alternate symbols does not in itself constitute deviation from the standard as long as *ampersand* is the concept associated with the character. Note that this does not necessarily restrict the use of such an alternate symbol design to mean "and"; in any type style *ampersand* may, of course, be used with arbitrary meaning.

(2) A card punch in one application may "skip" when the character HT (Horizontal Tabulation: used as skip) is presented to it; in another application the character HT may be recorded in the card without further action.

### B3. Related Larger and Smaller Sets

Consideration has been given to the relationship between the standard set and sets of other sizes. A number of straightforward logical transformations are possible which result in a variety of sets related to the standard. None of the transformed sets are recognized by this standard.

### B4. International Considerations

This standard conforms to the anticipated recommendations of the International Organization for Standardization (ISO) and the International Telegraph and Telephone Consultative Committee (CCITT)\* for a 7-bit code. It includes all the character assignments specified by those bodies for international standardization. Their recommendations, however, permit national standardization by the various countries in seven code table positions. Also, the characters in three additional positions have been designated as being replaceable by national characters in only those countries having an extraordinary requirement in this regard.

\*An international body which establishes standards and conventions for international telecommunications, especially where the public telegraph and telephone services are governmentally owned and operated. Their recommendations are often embodied in the regulations applying to such services.

The seven national usage positions and their assignments in this standard are shown in the following, as well as the three "supplementary" positions, which are shown in parentheses:

Column/Row	Character (U.S.)
4/0	@
5/11	[
5/12	\
5/13	]
(5/14)	'
(6/0)	'
7/11	{
7/12	:
7/13	}
(7/14)	~

In international interchange of information these 10 characters should not be used except where it is determined that there is agreement between sender and recipient.

In addition, in other countries, the number sign (#) (in position 2/3) may be replaced by "£".

### B5. Communications Considerations

Certain control characters are designated as "communication controls." They are:

SOH	(Start of Heading)
STX	(Start of Text)
ETX	(End of Text)
EOT	(End of Transmission)
ENQ	(Enquiry)
ACK	(Acknowledge)
DLE	(Data Link Escape)
NAK	(Negative Acknowledge)
SYN	(Synchronous Idle)
ETB	(End of Transmission Block)

These may be used by communication systems for their internal signaling, or for the exchange of information relating to the control of the communication system between that system and its end terminals. Some such systems may impose restrictions on the use of these communication control characters by the end terminals. For example, the use of some of them may be completely prohibited while others may be restricted to use in conformity with the formats and procedures required by the communication system for its operation.

## Appendix C

### Original Criteria

#### C1. Introduction

**C1.1** This Appendix contains the original criteria upon which the design of the code was based. Not all criteria have been entirely satisfied. Some are conflicting, and

the characteristics of the set represent accepted compromises of these divergent criteria.

**C1.2** The criteria were drawn from communication, processing, and media recording aspects of information interchange.

## C2. Criteria

- C2.1** Every character of the code set shall be represented by the same number of bits (i.e., binary digits).
- C2.2** The standard set shall be so structured as to facilitate derivation of logically related larger or smaller sets.
- C2.3** In a code of  $n$  bits, all possible  $2^n$  patterns of ones and zeros will be permitted and considered valid.
- C2.4** The number of bits,  $n$ , shall be sufficient to provide for the alphabetic and numeric characters, commonly used punctuation marks, and other special symbols, along with those control characters required for interchange of information.
- C2.5** The numerals 0 through 9 shall be included within a 4-bit subset.
- C2.6** The numerals 0 through 9 shall be so represented that the four low-order bits shall be the binary-coded-decimal form of the particular numeral that the code represents. In the selection of the two characters immediately succeeding the numeral 9, consideration shall be given to their replacement by the graphics 10 and 11 to facilitate the adoption of the code in the sterling monetary area.
- C2.7** The interspersion of control characters among the graphic characters shall be avoided. The characters devoted to controls shall be easily separable from those devoted to graphics.
- C2.8** Within the standard set, each character shall stand by itself and not depend on surrounding characters for interpretation.
- C2.9** An entire case of the alphabet (A through Z) shall be included within a 5-bit subset. Consideration shall be given to the need for more than 26 characters in some alphabets.
- C2.10** The letters of each case of the alphabet shall be

assigned, in conventional order (A through Z), to successive, increasing binary representations.

- C2.11** Suitable control characters required for communication and information processing shall be included.
- C2.12** Escape functions that provide for departures from the standard set shall be incorporated.
- C2.13** A simple binary comparison shall be sufficient to determine the order within each class of characters. (In this regard, the special graphics, the numerals, and the alphabet are each defined as distinct classes.) Simple binary rules do not necessarily apply between classes when ordering information.
- C2.14** Space (i.e., the space between words) must collate ahead of all other graphics.
- C2.15** Special symbols used in the ordering of information must collate ahead of both the alphabet and the numerals.
- C2.16** Insofar as possible, the special symbols shall be grouped according to their functions; for example, punctuation and mathematical symbols. Further, the set shall be so organized that the simplest possible test shall be adequate to distinguish and identify the basic alphabetic, numeric, and special symbol subsets.
- C2.17** Special symbols shall be placed in the set so as to simplify their generation by typewriters and similar keyboard devices. This criterion means, in effect, that the codes for pairs of characters that normally appear on the same keytops on a typewriter shall differ only in a common single-bit position.
- C2.18** The set shall contain the graphic characters of the principal programming languages.
- C2.19** The codes for all control characters shall contain a common, easily recognizable, bit pattern.
- C2.20** The Null (000...) and Delete (111...) characters shall be provided.

## Appendix D

### Terminology

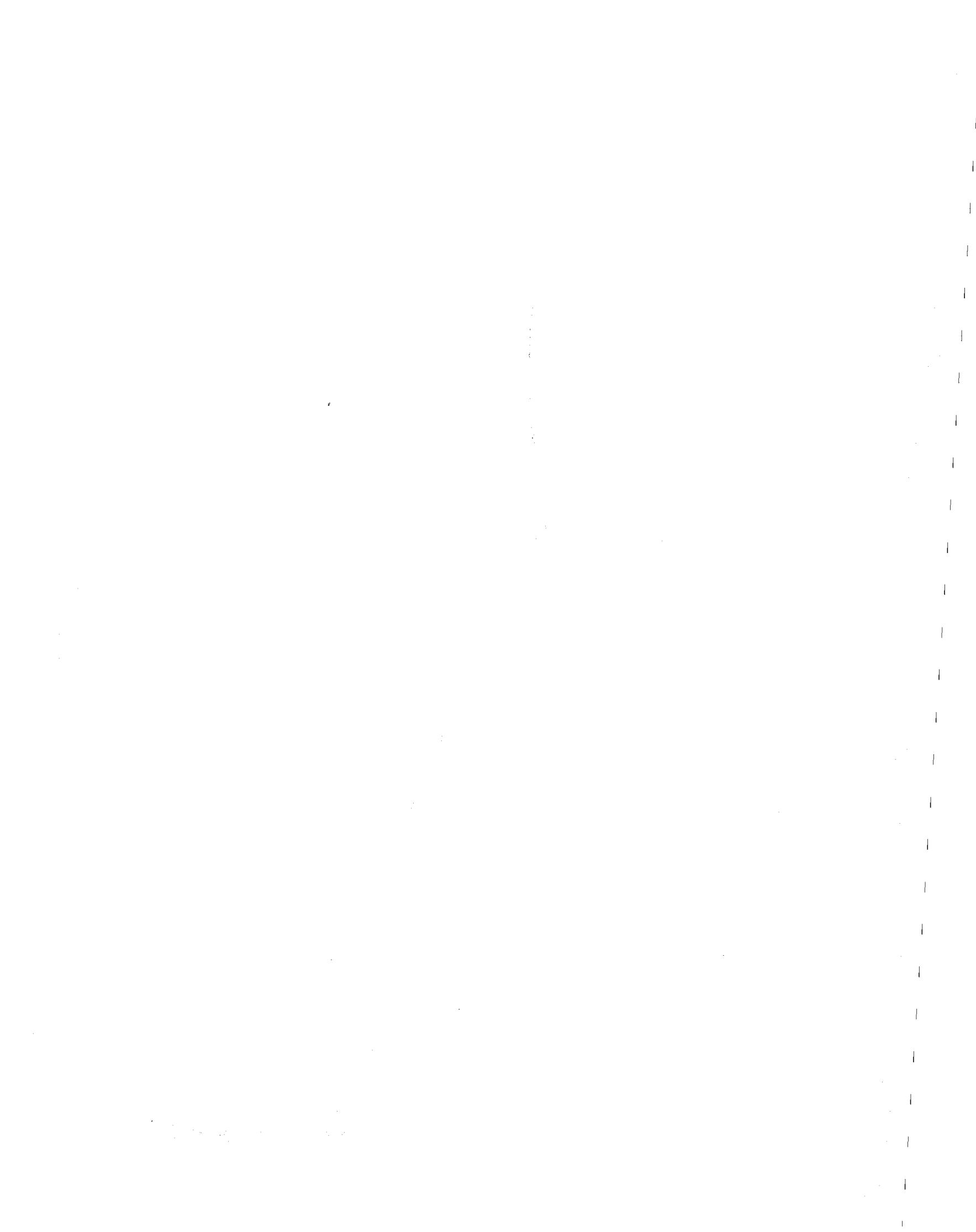
This Appendix is intended to clarify the sense in which certain terms are used.

**Bit:** Contraction of "binary digit."

**Bit Pattern:** The binary representation of a character.

**Character:** A member of a coded character set; the binary representation of such a member and its graphic symbol or control function.

**Code:** A system of discrete representation of a set of symbols and functions.



**FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)**

Nancy J. Neigus  
Bolt Beranek and Newman, Inc.  
Cambridge, Mass.

See Also: RFCs 354, 454, 495

**FILE TRANSFER PROTOCOL FOR THE ARPA NETWORK.**

---

**Preceding page blank**



FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

PREFACE

This document is the result of several months discussion via RFC (relevant numbers are 430, 448, 454, 463, 468, 478, 480), followed by a meeting of the FTP committee at BBN on March 16, followed by further communication among committee members. There are a considerable number of changes for the last "official" version, see RFCs 354, 385, but the gross structure remains the same. The places to look for differences are (1) in the definitions of types and modes, (2) in the specification of the data connection and data sockets, (3) in the command-reply sequences, (4) in the functions dependent on the TELNET protocol (FTP has been altered to correspond to the new TELNET spec). The model has been clarified and enlarged to allow inter-server file transfer, and several new commands have been added to accommodate more specialized (or site-specific) functions. It is my belief that this new specification reflects the views expressed by the committee at the above-mentioned meeting and in subsequent conversations.

The large number of incompatibilities would complicate a phased implementation schedule, such as is in effect for the TELNET protocol. Therefore we have assigned a new socket, decimal 21, as a temporary logger socket for the new version and a change-over date of 1 February 1974. Until that date the old (354, 385) version of FTP will be available on Socket 3 and the new version (attached) should be implemented on Socket 21. On 1 February the new version will shift to Socket 3 and the old disappear from view.

The File Transfer protocol should be considered stable at least until February, though one should feel free to propose further changes via RFC. (Implementation of new commands on an experimental basis is encouraged and should also be reported by RFC.) In addition, members of the FTP committee may be contacted directly about changes. Based on attendance at the March 16 meeting, they are:

Abhay Bhushan MIT-DMCG  
Bob Braden UCLA-CCN  
Bob Bressler BBN-NET  
Bob Clements BBN-TENEX  
John Day ILL-ANTS  
Peter Deutsch PARC-MAXC  
Wayne Hathaway AMES-67  
Mike Kudlick SRI-ARC  
Alex McKenzie BBN-NET  
Bob Merryman UCSD-CC  
Nancy Neigus BBN-NET  
Mike Padlipsky MIT-Multics  
Jim Pepin USC-44  
Ken Pogran MIT-Multics  
Jon Postel UCLA-NMC  
Milton Reese FNWC  
Brad Reussow HARV-10  
Marc Seriff MIT-DMCG  
Ed Taft HARV-10  
Bob Thomas BBN-TENEX  
Ric Werme CMU-10  
Jim White SRI-ARC

I would especially like to thank Bob Braden, Ken Pogran, Wayne Hathaway, Jon Postel, Ed Taft and Alex McKenzie for their help in preparing this document. NJN/jm

Preceding page blank



**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

**FILE TRANSFER PROTOCOL**

**INTRODUCTION**

The File Transfer Protocol (FTP) is a protocol for file transfer between Hosts (including Terminal Interface Message Processors (TIPs)) on the ARPA Computer Network (ARPANET). The primary function of FTP is to transfer files efficiently and reliably among Hosts and to allow the convenient use of remote file storage capabilities.

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among Hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs.

The attempt in this specification is to satisfy the diverse needs of users of maxi-Hosts, mini-Hosts, TIPs, and the Datacomputer, with a simple, and easily implemented protocol design.

This paper assumes knowledge of the following protocols described in NIC #7104:

The Host-Host Protocol

The Initial Connection Protocol

The TELNET Protocol

**DISCUSSION**

In this section, the terminology and the FTP model are discussed. The terms defined in this section are only those that have special significance in FTP.

**TERMINOLOGY**

**ASCII**

The USASCII character set as defined in NIC #7104. In FTP, ASCII characters are defined to be the lower half of an eight-bit code set (i.e., the most significant bit is zero).

**access controls**

Access controls define users' access privileges to the use of a system, and to the files in that system. Access controls are necessary to prevent unauthorized or accidental use of files. It is the prerogative of a server-FTP process to provide access controls.

**byte size**

The byte size specified for the transfer of data. The data connection is opened with this byte size. The data connection byte size is not necessarily the byte size in which data is to be stored in a system, nor the logical byte size for interpretation of the structure of the data.

**Preceding page blank**

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

data connection

A simplex connection over which data is transferred, in a specified byte size, mode and type. The data transferred may be a part of a file, an entire file or a number of files. The path may be between a server-DTP and a user-DTP, or between two server-DTPs.

data socket

The passive data transfer process "listens" on the data socket for an RFC from the active transfer process (server) in order to open the data connection. The server has fixed data sockets; the passive process may or may not.

EOF

The end-of-file condition that defines the end of a file being transferred.

EOR

The end-of-record condition that defines the end of a record being transferred.

error recovery

A procedure that allows a user to recover from certain errors such as failure of either Host system or transfer process. In FTP, error recovery may involve restarting a file transfer at a given checkpoint.

FTP commands

A set of commands that comprise the control information flowing from the user-FTP to the server-FTP process.

file

An ordered set of computer data (including programs), of arbitrary length, uniquely identified by a pathname.

mode

The mode in which data is to be transferred via the data connection. The mode defines the data format during transfer including EOR and EOF. The transfer modes defined in FTP are described in the Section on Transmission Modes.

NVT

The Network Virtual Terminal as defined in the ARPANET TELNET Protocol.

NVFS

The Network Virtual File System. A concept which defines a standard network file system with standard commands and pathname conventions. FTP only partially embraces the NVFS concept at this time.

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

pathname

Pathname is defined to be the character string which must be input to a file system by a user in order to identify a file. Pathname normally contains device and/or directory names, and file name specification. FTP does not yet specify a standard pathname convention. Each user must follow the file naming conventions of the file systems he wishes to use.

record

A sequential file may be structured as a number of contiguous parts called records. Record structures are supported by FTP but a file need not have record structure.

reply

A reply is an acknowledgment (positive or negative) sent from server to user via the TELNET connections in response to FTP commands. The general form of a reply is a completion code (including error codes) followed by a text string. The codes are for use by programs and the text is usually intended for human users.

server-DTP

The data transfer process, in its normal "active" state, establishes the data connection by RFC to the "listening" data socket, sets up parameters for transfer and storage, and transfers data on command from its PI. The DTP can be placed in a "passive" state to listen for, rather than initiate, an RFC on the data socket.

server-FTP process

A process or set of processes which perform the function of file transfer in cooperation with a user-FTP process and, possibly, another server. The functions consist of a protocol interpreter (PI) and a data transfer process (DTP).

server-PI

The protocol interpreter "listens" on Socket 3 for an ICP from a user-PI and establishes a TELNET communication connection. It receives standard FTP commands from the user-PI, sends replies, and governs the server-DTP.

TELNET connections

The full-duplex communication path between a user-PI and a server-PI. The TELNET connections are established via the standard ARPANET Initial Connection Protocol (ICP).

type

The data representation type used for data transfer and storage. Type implies certain transformations between the time of data storage and data transfer. The representation types defined in FTP are described in the Section on Establishing Data Connections.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

user

A human being or a process on behalf of a human being wishing to obtain file transfer service. The human user may interact directly with a server-FTP process, but use of a user-FTP process is preferred since the protocol design is weighted towards automata.

user-DTP

The data transfer process "listens" on the data socket for an RFC from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive.

user-FTP process

A set of functions including a protocol interpreter, a data transfer process and a user interface which together perform the function of file transfer in cooperation with one or more server-FTP processes. The user interface allows a local language to be used in the command-reply dialogue with the user.

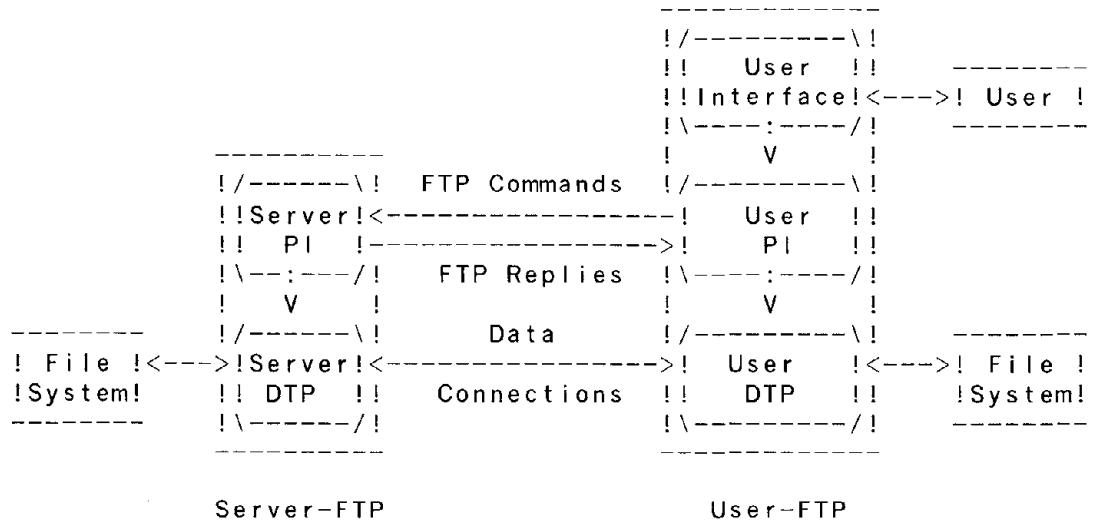
user-PI

The protocol interpreter initiates the ICP to the server-FTP process, initiates FTP commands, and governs the user-DTP if that process is part of the file transfer.

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

### THE FTP MODEL

With the above definitions in mind, the following model (shown in Figure 1) may be diagrammed for an FTP service.



- NOTES: 1. The data connection may be in either direction.  
2. The data connection need not exist all of the time.

Figure 1 Model for FTP Use

In the model described in Figure 1, the user-protocol interpreter initiates the TELNET connections. At the initiation of the user, standard FTP commands are generated by the user-PI and transmitted to the server process via the TELNET connections. (The user may establish a direct TELNET connection to the server-FTP, from a TIP terminal for example, and generate standard FTP commands himself, by-passing the user-FTP process.) Standard replies are sent from the server-PI to the user-PI over the TELNET connections in response to the commands.

The FTP commands specify the parameters for the data connection (data socket, byte size, transfer mode, representation type, and structure) and the nature of file system operation (store, retrieve, append, delete, etc.). The user-DTP or its designate should "listen" on the specified data socket, and the server initiate the data connection and data transfer in accordance with the specified parameters. It should be noted that the data socket need not be in the same Host that initiates the FTP commands via the TELNET connections, but the user or his user-FTP process must ensure a "listen" on the specified data socket. It should also be noted that two data connections, one for send and the other for receive, may exist simultaneously.

In another situation a user might wish to transfer files between two Hosts, neither of which is his local Host. He sets up TELNET connections to the two servers and then arranges for a data connection between them. In this manner control information is passed to the user-PI but data is transferred between the server data transfer processes. Following is a model of this server-server interaction.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

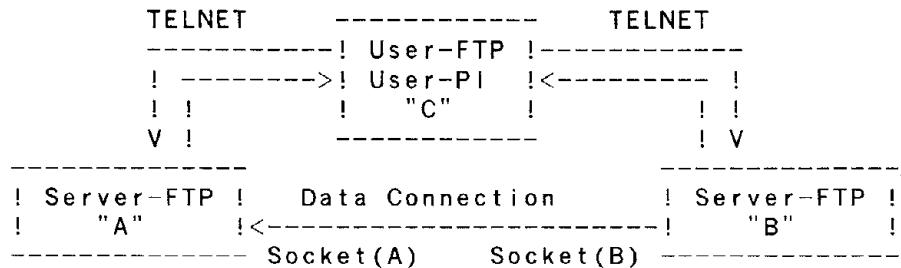


Figure 2

The protocol requires that the TELNET connections be open while data transfer is in progress. It is the responsibility of the user to request the closing of the TELNET connections when finished using the FTP service, while it is the server who takes the action. The server may abort data transfer if the TELNET connections are closed without command.

#### DATA TRANSFER FUNCTIONS

Files are transferred only via the data connection(s). The TELNET connection is used for the transfer of commands, which describe the functions to be performed, and the replies to these commands (see the Section on FTP Replies). Several commands are concerned with the transfer of data between Hosts. These data transfer commands include the BYTE, MODE, and SOCKET commands which specify how the bits of the data are to be transmitted, and the STRUCTure and TYPE commands, which are used to define the way in which the data are to be represented. The transmission and representation are basically independent but "Stream" transmission mode is dependent on the file structure attribute and if "Compressed" transmission mode is used the nature of the filler byte depends on the representation type.

#### DATA REPRESENTATION AND STORAGE

Data is transferred from a storage device in the sending Host to a storage device in the receiving Host. Often it is necessary to perform certain transformations on the data because data storage representations in the two systems are different. For example, NVT-ASCII has different data storage representations in different systems. PDP-10's generally store NVT-ASCII as five 7-bit ASCII characters, left-justified in a 36-bit word. 360's store NVT-ASCII as 8-bit EBCDIC codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit word. It may be desirable to convert characters into the standard NVT-ASCII representation when transmitting text between dissimilar systems. The sending and receiving sites would have to perform the necessary transformations between the standard representation and their internal representations.

A different problem in representation arises when transmitting binary data (not character codes) between Host systems with different word lengths. It is not always clear how the sender should send data, and the receiver store it. For example, when transmitting 32-bit bytes from a 32-bit word-length system to a 36-bit word-length system, it may be desirable (for reasons of efficiency and usefulness) to store the 32-bit bytes right-justified in a 36-bit word in the latter system. In any case, the user should have the option of specifying data representation and transformation functions. It should be noted that FTP provides for very limited data type representations. Transformations desired beyond this

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

limited capability should be performed by the user directly or via the use of the Data Reconfiguration Service (DRS, RFC #138, NIC #6715). Additional representation types may be defined later if there is a demonstrable need.

Data representations are handled in FTP by a user specifying a representation type. This type may implicitly (as in ASCII or EBCDIC) or explicitly (as in Local byte) define a byte size for interpretation which is referred to as the "logical byte size." This has nothing to do with the byte size used for transmission over the data connection(s) (called the "transfer byte size") and the two should not be confused. For example, NVT-ASCII has a logical byte size of 8 bits but an ASCII file might be transferred using a transfer byte size of 32. If the type is Local byte, then the TYPE command has an obligatory second parameter specifying the logical byte size.

The types ASCII and EBCDIC also take a second (optional) parameter; this is to indicate what kind of vertical format control, if any, is associated with a file. The following data representation types are defined in FTP:

#### ASCII Format

This is the default type and must be accepted by all FTP implementations. It is intended primarily for the transfer of text files, except when both Hosts would find the EBCDIC type more convenient.

The sender converts the data from his internal character representation to the standard 8-bit NVT-ASCII representation (see the TELNET specification). The receiver will convert the data from the standard form to his own internal form.

In accordance with the NVT standard, the <CRLF> sequence should be used, where necessary, to denote the end of a line of text. (See the discussion of file structure at the end of the Section on Data Representation and Storage).

Using the standard NVT-ASCII representation means that data must be interpreted as 8-bit bytes. If the BYTE command (see the Section on Transfer Parameter Commands) specifies a transfer byte size different from 8 bits, the 8-bit ASCII characters should be packed contiguously without regard for transfer byte boundaries.

The Format parameter for ASCII and EBCDIC types is discussed below.

#### EBCDIC Format

This type is intended for efficient transfer between Hosts which use EBCDIC for their internal character representation.

For transmission the data are represented as 8-bit EBCDIC characters. The character code is the only difference between the functional specifications of EBCDIC and ASCII types.

End-of-line (as opposed to end-of-record--see the discussion of structure) will probably be rarely used with EBCDIC type for purposes of denoting structure, but where it is necessary the <NL> character should be used.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

A character file may be transferred to a Host for one of three purposes: for printing, for storage and later retrieval, or for processing. If a file is sent for printing, the receiving Host must know how the vertical format control is represented. In the second case, it must be possible to store a file at a Host and then retrieve it later in exactly the same form. Finally, it ought to be possible to move a file from one Host to another and process the file at the second Host without undue trouble. A single ASCII or EBCDIC format does not satisfy all these conditions and so these types have a second parameter specifying one of the following three formats:

Non-print

This is the default format to be used if the second (format) parameter is omitted. Non-print format must be accepted by all FTP implementations.

The file need contain no vertical format information. If it is passed to a printer process, this process may assume standard values for spacing and margins.

Normally, this format will be used with files destined for processing or just storage.

TELNET Format Controls

The file contains ASCII/EBCDIC vertical format controls (i.e., <CR>, <LF>, <NL>, <VT>, <FF>) which the printer process will interpret appropriately. <CRLF>, in exactly this sequence, also denotes end-of-line.

Carriage Control (ASA)

The file contains ASA (FORTRAN) vertical format control characters. (See NWG/RFC #189 Appendix C and Communications of the ACM, Vol. 7, No. 10, 606 (Oct. 1964)). In a line or a record, formatted according to the ASA Standard, the first character is not to be printed. Instead it should be used to determine the vertical movement of the paper which should take place before the rest of the record is printed. The ASA Standard specifies the following control characters:

Character	Vertical Spacing
blank	Move paper up one line
0	Move paper up two lines
1	Move paper to top of next page
+	No movement, i.e., overprint

Clearly there must be some way for a printer process to distinguish the end of the structural entity. If a file has record structure (see below) this is no problem; records will be explicitly marked during transfer and storage. If the file has no record structure, the <CRLF> end-of-line sequence is used to separate printing lines, but these format effectors are overridden by the ASA controls.

Image

The data are sent as contiguous bits which, for transfer, are packed into transfer bytes of the size specified in the BYTE command. The receiving site must store the data as contiguous bits. The structure of the storage system might necessitate the padding of

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

the file (or of each record, for a record-structured file) to some convenient boundary (byte, word or block). This padding, which must be all zeroes, may occur only at the end of the file (or at the end of each record) and there must be a way of identifying the padding bits so that they may be stripped off if the file is retrieved. The padding transformation should be well publicized to enable a user to process a file at the storage site.

Image type is intended for the efficient storage and retrieval of files and for the transfer of binary data. It is recommended that this type be accepted by all FTP implementations.

**Local byte Byte size**

The data is transferred in logical bytes of the size specified by the obligatory second parameter, Byte size. The value of Byte size must be a decimal integer; there is no default value. The logical byte size is not necessarily the same as the transfer byte size. If there is a difference in byte sizes, then the logical bytes should be packed contiguously, disregarding transfer byte boundaries and with any necessary padding at the end.

When the data reaches the receiving Host it will be transformed in a manner dependent on the logical byte size and the particular Host. This transformation must be invertible (that is an identical file can be retrieved if the same parameters are used) and should be well publicized by the FTP implementors.

This type is intended for the transfer of structured data. For example, a user sending 36-bit floating-point numbers to a Host with a 32-bit word could send his data as Local byte with a logical byte size of 36. The receiving Host would then be expected to store the logical bytes so that they could be easily manipulated; in this example putting the 36-bit logical bytes into 64-bit double words should suffice.

A note of caution about parameters: a file must be stored and retrieved with the same parameters if the retrieved version is to be identical to the version originally transmitted. Conversely, FTP implementations must return a file identical to the original if the parameters used to store and retrieve a file are the same.

In addition to different representation types, FTP allows the structure of a file to be specified. Currently two file structures are recognized in FTP: file-structure, where there is no internal structure, and record-structure, where the file is made up of records. File-structure is the default, to be assumed if the STRUCTURE command has not been used but both structures must be accepted for "text" files (i.e., files with TYPE ASCII or EBCDIC) by all FTP implementations. The structure of a file will affect both the transfer mode of a file (see the Section on Transmission Modes) and the interpretation and storage of the file.

The "natural" structure of a file will depend on which Host stores the file. A source-code file will usually be stored on an IBM 360 in fixed length records but on a PDP-10 as a stream of characters partitioned into lines, for example by <CRLF>. If the transfer of files between such disparate sites is to be useful, there must be some way for one site to recognize the other's assumptions about the file.

## FILE TRANSFER PROTOCOL

RFC 542 NIC 17759 (Aug. 12, 1973)

With some sites being naturally file-oriented and others naturally record-oriented there may be problems if a file with one structure is sent to a Host oriented to the other. If a text file is sent with record-structure to a Host which is file oriented, then that Host should apply an internal transformation to the file based on the record structure. Obviously this transformation should be useful but it must also be invertible so that an identical file may be retrieved using record structure.

In the case of a file being sent with file-structure to a record-oriented Host, there exists the question of what criteria the Host should use to divide the file into records which can be processed locally. If this division is necessary the FTP implementation should use the end-of-line sequence, <CRLF> for ASCII, or <NL> for EBCDIC text files, as the delimiter. If an FTP implementation adopts this technique, it must be prepared to reverse the transformation if the file is retrieved with file-structure.

### ESTABLISHING DATA CONNECTIONS

The mechanics of transferring data consists of setting up the data connection to the appropriate sockets and choosing the parameters for transfer--byte size and mode. Both the user and the server-DTPs have default data sockets; these are the two sockets (for send and receive) immediately following the standard ICP TELNET socket ,i.e., (U+4) and (U+5) for the user-process and (S+2), (S+3) for the server. The use of default sockets will ensure the security of the data transfer, without requiring the socket information to be explicitly exchanged.

The byte size for the data connection is specified by the BYTE command, or, if left unspecified, defaults to 8-bit bytes. This byte size is relevant only for the actual transfer of the data; it has no bearing on representation of the data within a Host's file system. The protocol does not require servers to accept all possible byte sizes. Since the use of various byte sizes is intended for efficiency of transfer, servers may implement only those sizes for which their data transfer is efficient including the default byte size of 8 bits.

The passive data transfer process (this may be a user-DTP or a second server-DTP) shall "listen" on the data socket prior to sending a transfer request command. The FTP request command determines the direction of the data transfer and thus which data socket (odd or even) is to be used in establishing the connection. The server, upon receiving the transfer request, will initiate the data connection by RFC to the appropriate socket using the specified (or default) byte size. When the connection is opened, the data transfer begins between DTP's, and the server-PI sends a confirming reply to the user-PI.

It is possible for the user to specify an alternate data socket by use of the SOCK command. He might want a file dumped on a TIP line printer or retrieved from a third party Host. In the latter case the user-PI sets up TELNET connections with both server-PI's and sends each a SOCK command indicating the fixed data sockets of the other. One server is then told (by an FTP command) to "listen" for an RFC which the other will initiate and finally both are sent the appropriate transfer commands. The exact sequence of commands and replies sent between the user-controller and the servers is defined in the Section on FTP Replies.

In general it is the server's responsibility to maintain the data connection--to initiate the RFC's and the closes. The exception to this is when the user-DTP is sending the data in a transfer mode that requires the connection to be closed to indicate EOF. The server MUST close the data connection under the following conditions:

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

1. The server has completed sending data in a transfer mode that requires a close to indicate EOF.
2. The server receives an ABORT command from the user.
3. The socket or byte size specification is changed by a command from the user.
4. The TELNET connections are closed legally or otherwise.
5. An irrecoverable error condition occurs.

Otherwise the close is a server option, the exercise of which he must indicate to the user-process by an appropriate reply.

#### TRANSMISSION MODES

The next consideration in transferring data is choosing the appropriate transmission mode. There are three modes: one which formats the data and allows for restart procedures; one which also compresses the data for efficient transfer; and one which passes the data with little or no processing. In this last case the mode interacts with the structure attribute to determine the type of processing. In the compressed mode the representation type determines the filler byte.

All data transfers must be completed with an end-of-file (EOF) which may be explicitly stated or implied by the closing of the data connection. For files with record structure, all the end-of-record markers (EOR) are explicit, including the final one.

NOTE: In the rest of this section, byte means "transfer byte" except where explicitly stated otherwise.

The following transmission modes are defined in FTP:

#### STREAM

The data is transmitted as a stream of bytes. There is no restriction on the representation type used; record structures are allowed, in which case the transfer byte size must be at least 3 bits!

In a record structured file EOR and EOF will each be indicated by a two-byte control code of whatever byte size is used for the transfer. The first byte of the control code will be all ones, the escape character. The second byte will have the low order bit on and zeroes elsewhere for EOR and the second low order bit on for EOF; that is, the byte will have value 1 for EOR and value 2 for EOF. EOR and EOF may be indicated together on the last byte transmitted by turning both low order bits on, i.e., the value 3. If a byte of all ones was intended to be sent as data, it should be repeated in the second byte of the control code.

If the file does not have record structure, the EOF is indicated by the sending Host closing the data connection and all bytes are data bytes.

For the purpose of standardized transfer, the sending Host will translate his internal end of line or end of record denotation into the representation prescribed by the transfer mode and

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

file structure, and the receiving Host will perform the inverse translation to his internal denotation. An IBM 360 record count field may not be recognized at another Host, so the end of record information may be transferred as a two byte control code in Stream mode or as a flagged bit in a Block or Compressed mode descriptor. End of line in an ASCII or EBCDIC file with no record structure should be indicated by <CRLF> or <NL>, respectively. Since these transformations imply extra work for some systems, identical systems transferring non-record structured text files might wish to use a binary representation and stream mode for the transfer.

BLOCK

The file is transmitted as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field, and descriptor code. The count field indicates the total length of the data block in bytes, thus marking the beginning of the next data block (there are no filler bits). The descriptor code defines: last block in the file (EOF) last block in the record (EOR), restart marker (see the Section on Error Recovery and Restart) or suspect data (i.e., the data being transferred is suspected of errors and is not reliable). This last code is NOT intended for error control within FTP. It is motivated by the desire of sites exchanging certain types of data (e.g., seismic or weather data) to send and receive all the data despite local errors (such as "magnetic tape read errors"), but to indicate in the transmission that certain portions are suspect). Record structures are allowed in this mode, and any representation type may be used. There is no restriction on the transfer byte size.

The header consists of the smallest integral number of bytes whose length is greater than or equal to 24 bits. Only the LEAST significant 24 bits (right-justified) of header shall have information; the remaining most significant bits are "don't care" bits. Of the 24 bits of header information, the 16 low order bits shall represent byte count, and the 8 high order bits shall represent descriptor codes as shown below.

Integral number of bytes greater than or equal to 24 bits

!	Don't care	!	Descriptor	!	Byte Count	!
!	0 to 231 bits	!	8 bits	!	16 bits	!

The descriptor codes are indicated by bit flags in the descriptor byte. Four codes have been assigned, where each code number is the decimal value of the corresponding bit in the byte.

Code	Meaning
128	End of data block is EOR
64	End of data block is EOF
32	Suspected errors in data block
16	Data block is a restart marker

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

With this encoding more than one descriptor coded condition may exist for a particular block. As many bits as necessary may be flagged.

The restart marker is embedded in the data stream as an integral number of 8-bit bytes representing printable characters in the language being used over the TELNET connection (e.g., default--NVT-ASCII). These marker bytes are right-justified in the smallest integral number of transfer bytes greater than or equal to 8 bits. For example, if the byte size is 7 bits, the restart marker byte would be one byte right-justified per two 7-bit bytes as shown below:

Two 7-bit bytes

```
-----  
!           ! Marker Char !  
!           !     8 bits   !  
-----
```

If the transfer byte size is 16 or more bits, the maximum possible number of complete marker bytes should be packed, right-justified, into each transfer byte. The restart marker should begin in the first marker byte. If there are any unused marker bytes, these should be filled with the character <SP> (Space, in the appropriate language). <SP> must not be used WITHIN a restart marker. For example, to transmit a six-character marker with a 36-bit transfer byte size, the following three 36-bit bytes would be sent:

```
-----  
! Don't care !Descriptor! Byte count = 2 !  
!    12 bits  ! code = 16!  
-----
```

```
-----  
!      ! Marker ! Marker ! Marker ! Marker !  
!      ! 8 bits ! 8 bits ! 8 bits ! 8 bits !  
-----
```

```
-----  
!      ! Marker ! Marker ! Space   ! Space   !  
!      ! 8 bits ! 8 bits ! 8 bits ! 8 bits !  
-----
```

### Compressed

The file is transmitted as series of bytes of the size specified by the BYTE command. There are three kinds of information to be sent: regular data, sent in a byte string; compressed data, consisting of replications or filler; and control information, sent in a two-byte escape sequence. If the byte size is B bits and n>0 bytes of regular data are sent, these n bytes are preceded by a byte with the left-most bit set to 0 and the right-most B-1 bits containing the number n.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

1	B-1	B	B
----- ----- -----			
Byte string:		!0! n ! !d(1)!...!d(n)!	
----- ----- -----			
!---n bytes---! of data			

String of n data bytes d(1),..., d(n)  
Count n must be positive.

To compress a string of n replications of the data byte d, the following 2 bytes are sent:

2	B-2	B
----- ----- -----		
Replicated Byte: ! 1 0 ! n ! ! d !		
----- ----- -----		

A string of n filler bytes can be compressed into a single byte, where the filler byte varies with the representation type. If the type is ASCII or EBCDIC the filler byte is <SP> (Space, ASCII code 32., EBCDIC code 64). If the transfer byte size is not 8, the expanded byte string should be filled with 8-bit <SP> characters in the manner described in the definition of ASCII representation type (see the Section on Data Representation and Storage). If the type is Image or Local byte the filler is a zero byte.

2	B-2	
-----		
Filler String: ! 1 1 ! n !		
-----		

The escape sequence is a double byte, the first of which is the escape byte (all zeroes) and the second of which contains descriptor codes as defined in Block mode. This implies that the byte size must be at least 8 bits, which is not much of a restriction for efficiency in this mode. The descriptor codes have the same meaning as in Block mode and apply to the succeeding string of bytes.

Compressed mode is useful for obtaining increased bandwidth on very large network transmissions at a little extra CPU cost. It is most efficient when the byte size chosen is that of the word size of the transmitting Host, and can be most effectively used to reduce the size of printer files such as those generated by RJE Hosts.

#### ERROR RECOVERY AND RESTART

There is no provision for detecting bits lost or scrambled in data transfer. This issue is perhaps handled best at the NCP level where it benefits most users. However, a restart procedure is provided to protect users from gross system failures (including failures of a Host, an FTP-process, or the IMP subnet).

The restart procedure is defined only for the block and compressed modes of data transfer. It requires the sender of data to insert a special marker code in the data stream with some marker information. The marker information has meaning only to the sender, but must consist

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

of printable characters in the default or negotiated language of the TELNET connection. The marker could represent a bit-count, a record-count, or any other information by which a system may identify a data checkpoint. The receiver of data, if it implements the restart procedure, would then mark the corresponding position of this marker in the receiving system, and return this information to the user.

In the event of a system failure, the user can restart the data transfer by identifying the marker point with the FTP restart procedure. The following example illustrates the use of the restart procedure.

The sender of the data inserts an appropriate marker block in the data stream at a convenient point. The receiving Host marks the corresponding data point in its file system and conveys the last known sender and receiver marker information to the user, either directly or over the TELNET connection in a 251 reply (depending on who is the sender). In the event of a system failure, the user or controller process restarts the server at the last server marker by sending a restart command with server's marker code as its argument. The restart command is transmitted over the TELNET connection and is immediately followed by the command (such as RETR, STOR or LIST) which was being executed when the system failure occurred.

#### FILE TRANSFER FUNCTIONS

The communication channel from the user-PI to the server-PI is established by ICP from the user to a standard server socket. The user protocol interpreter is responsible for sending FTP commands and interpreting the replies received; the server-PI interprets commands, sends replies and directs its DTP to set up the data connection and transfer the data. If the second party to the data transfer (the passive transfer process) is the user-DTP then it is governed through the internal protocol of the user-FTP Host; if it is a second server-DTP then it is governed by its PI on command from the user-PI.

#### FTP COMMANDS

The File Transfer Protocol follows the specifications of the TELNET protocol for all communications over the TELNET connection - see NIC #7104. Since, in the future, the language used for TELNET communication may be a negotiated option, all references in the next two sections will be to the "TELNET language" and the corresponding "TELNET end of line code". Currently one may take these to mean NVT-ASCII and <CRLF>. No other specifications of the TELNET protocol will be cited.

FTP commands are "TELNET strings" terminated by the "TELNET end of line code". The command codes themselves are alphabetic characters terminated by the character <SP> (Space) if parameters follow and TELNET-EOL otherwise. The command codes and the semantics of commands are described in this section; the detailed syntax of commands is specified in the Section on Commands, the reply sequences are discussed in the Section on Sequencing of Commands and Replies, and scenarios illustrating the use of commands are provided in the Section on Typical FTP Scenarios.

FTP commands may be partitioned as those specifying access-control identifiers, data transfer parameters, or FTP service requests. Certain commands (such as ABOR, STAT, BYE) may be sent over the TELNET connections while a data transfer is in progress. Some servers may not be able to monitor the TELNET and data connections simultaneously, in which case some

## FILE TRANSFER PROTOCOL

RFC 542 NIC 17759 (Aug. 12, 1973)

special action will be necessary to get the server's attention. The exact form of the "special action" is related to decisions currently under review by the TELNET committee; but the following ordered format is tentatively recommended:

1. User system inserts the TELNET "Interrupt Process" (IP) signal in the TELNET stream.
2. User system sends the TELNET "Synch" signal
3. User system inserts the command (e.g., ABOR) in the TELNET stream.
4. Server PI,, after receiving "IP", scans the TELNET stream for EXACTLY ONE FTP command.

(For other servers this may not be necessary but the actions listed above should have no unusual effect.)

### ACCESS CONTROL COMMANDS

The following commands specify access control identifiers (command codes are shown in parentheses).

#### USER NAME (USER)

The argument field is a TELNET string identifying the user. The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the TELNET connections are made (some servers may require this). Additional identification information in the form of a password and/or an account command may also be required by some servers. Servers may allow a new USER command to be entered at any point in order to change the access control and/or accounting information. This has the effect of flushing any user, password, and account information already supplied and beginning the login sequence again. All transfer parameters are unchanged and any file transfer in progress is completed under the old account.

#### PASSWORD (PASS)

The argument field is a TELNET string identifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress typeout. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user-FTP process to hide the sensitive password information.

#### ACCOUNT (ACCT)

The argument field is a TELNET string identifying the user's account. The command is not necessarily related to the USER command, as some sites may require an account for login and others only for specific access, such as storing files. In the latter case the command may arrive at any time. There are two reply codes to differentiate these cases for the automaton: when account information is required for login, the response to a successful PASSword command is reply code 331; then if a command other than ACCounT is sent, the server may remember it and return a 331 reply, prepared to act

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

on the command after the account information is received; or he may flush the command and return a 433 reply asking for the account. On the other hand, if account information is NOT required for login, the reply to a successful PASSword command is 230; and if the information is needed for a command issued later in the dialogue, the server should return a 331 or 433 reply depending on whether he stores (pending receipt of the ACCouNT command) or discards the command, respectively.

#### REINITIALIZE (REIN)

This command terminates a USER, flushing all I/O and account information, except to allow any transfer in progress to be completed. All parameters are reset to the default settings and the TELNET connection is left open. This is identical to the state in which a user finds himself immediately after the ICP is completed and the TELNET connections are opened. A USER command may be expected to follow.

#### LOGOUT (BYE)

This command terminates a USER and if file transfer is not in progress, the server closes the TELNET connection. If file transfer is in progress, the connection will remain open for result response and the server will then close it. If the user-process is transferring files for several USERS but does not wish to close and then reopen connections for each, then the REIN command should be used instead of BYE.

An unexpected close on the TELNET connection will cause the server to take the effective action of an abort (ABOR) and a logout (BYE).

### TRANSFER PARAMETER COMMANDS

All data transfer parameters have default values, and the commands specifying data transfer parameters are required only if the default parameter values are to be changed. The default value is the last specified value, or if no value has been specified, the standard default value as stated here. This implies that the server must "remember" the applicable default values. The commands may be in any order except that they must precede the FTP service request. The following commands specify data transfer parameters.

#### BYTE SIZE (BYTE)

The argument is a decimal integer (1 through 255) specifying the byte size for the data connection. The default byte size is 8 bits. A server may reject certain byte sizes that he has not implemented.

#### DATA SOCKET (SOCK)

The argument is a HOST-SOCKET specification for the data socket to be used in data connection. There may be two data sockets, one for transfer from the "active" DTP to the "passive" DTP and one for "passive" to "active". An odd socket number defines a send socket and an even socket number defines a receive socket. The default HOST is the user Host to which TELNET connections are made. The default data sockets are (U+4) and (U+5) where U is the socket number used in the TELNET ICP and the TELNET connections are on sockets (U+2) and (U+3). The server has fixed data sockets (S+2) and (S+3) as well, and under normal circumstances this command and its reply are not needed.

## FILE TRANSFER PROTOCOL

RFC 542 NIC 17759 (Aug. 12, 1973)

### PASSIVE (PASV)

This command requests the server-DTP to "listen" on both of his data sockets and to wait for an RFC to arrive for one socket rather than initiate one upon receipt of a transfer command. It is assumed the server has already received a SOCK command to indicate the foreign socket from which the RFC will arrive to ensure the security of the transfer.

### REPRESENTATION TYPE (TYPE)

The argument specifies the representation type as described in the Section on Data Representation and Storage. Several types take a second parameter. The first parameter is denoted by a single TELNET character, as is the second Format parameter for ASCII and EBCDIC; the second parameter for local byte is a decimal integer to indicate Bytesize. The parameters are separated by a <SP> (Space, ASCII code 32.). The following codes are assigned for type:

```
\   /  
A - ASCII !   ! N - Non-print  
           !-><-! T - TELNET format effectors  
E - EBCDIC!   ! C - Carriage Control (ASA)  
           /   \  
I - Image  
  
L # - Local byte Bytesize
```

The default representation type is ASCII Non-print. If the Format parameter is changed, and later just the first argument is changed, Format then returns to the Non-print default.

### FILE STRUCTURE (STRU)

The argument is a single TELNET character code specifying file structure described in the Section on Data Representation and Storage. The following codes are assigned for structure:

```
F - File (no record structure)  
R - Record structure
```

The default structure is File (i.e., no records).

### TRANSFER MODE (MODE)

The argument is a single TELNET character code specifying the data transfer modes described in the Section on Transmission Modes. The following codes are assigned for transfer modes:

```
S - Stream  
B - Block  
C - Compressed
```

The default transfer mode is Stream.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

#### FTP SERVICE COMMANDS

The FTP service commands define the file transfer or the file system function requested by the user. The argument of an FTP service command will normally be a pathname. The syntax of pathnames must conform to server site conventions (with standard defaults applicable), and the language conventions of the TELNET connection. The suggested default handling is to use the last specified device, directory or file name, or the standard default defined for local users. The commands may be in any order except that a "rename from" command must be followed by a "rename to" command and the restart command must be followed by the interrupted service command. The data, when transferred in response to FTP service commands, shall always be sent over the data connection, except for certain informative replies. The following commands specify FTP service requests:

##### RETRIEVE (RETR)

This command causes the server-DTP to transfer a copy of the file, specified in the pathname, to the server- or user-DTP at the other end of the data connection. The status and contents of the file at the server site shall be unaffected.

##### STORE (STOR)

This command causes the server-DTP to accept the data transferred via the data connection and to store the data as a file at the server site. If the file specified in the pathname exists at the server site then its contents shall be replaced by the data being transferred. A new file is created at the server site if the file specified in the pathname does not already exist.

##### APPEND (with create) (APPE)

This command causes the server-DTP to accept the data transferred via the data connection and to store the data in a file at the server site. If the file specified in the pathname exists at the server site, then the data shall be appended to that file; otherwise the file specified in the pathname shall be created at the server site.

##### ALLOCATE (ALLO)

This command may be required by some servers to reserve sufficient storage to accommodate the new file to be transferred. The argument shall be a decimal integer representing the number of bytes (using the logical byte size) of storage to be reserved for the file. For files sent with record structure a maximum record size (in logical bytes) might also be necessary; this is indicated by a decimal integer in a second argument field of the command. This second argument is optional, but when present should be separated from the first by the three TELNET characters <SP> R <SP>. This command shall be followed by a STORe or APPEnd command. The ALLO command should be treated as a NOOP (no operation) by those servers which do not require that the maximum size of the file be declared beforehand, and those servers interested in only the maximum record size should accept a dummy value in the first argument and ignore it.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

**RESTART (REST)**

The argument field represents the server marker at which file transfer is to be restarted. This command does not cause file transfer but "spaces" over the file to the specified data checkpoint. This command shall be immediately followed by the appropriate FTP service command which shall cause file transfer to resume.

**RENAME FROM (RNFR)**

This command specifies the file which is to be renamed. This command must be immediately followed by a "rename to" command specifying the new file pathname.

**RENAME TO (RNTO)**

This command specifies the new pathname of the file specified in the immediately preceding "rename from" command. Together the two commands cause a file to be renamed.

**ABORT (ABOR)**

This command indicates to the server to abort the previous FTP service command and any associated transfer of data. The abort command may require "special action", as discussed in the Section on FTP Commands, to force recognition by the server. No action is to be taken if the previous command has been completed (including data transfer). The TELNET connections are not to be closed by the server, but the data connection must be closed. An appropriate reply should be sent by the server in all cases.

**DELETE (DELE)**

This command causes the file specified in the pathname to be deleted at the server site. If an extra level of protection is desired (such as the query, "DO you really wish to delete?"), it should be provided by the user-FTP process.

**LIST (LIST)**

This command causes a list to be sent from the server to the passive DTP. If the pathname specifies a directory, the server should transfer a list of files in the specified directory. If the pathname specifies a file then the server should send current information on the file. A null argument implies the user's current working or default directory. The data transfer is over the data connection in type ASCII or type EBCDIC. (The user must ensure that the TYPE is appropriately ASCII or EBCDIC).

**NAME-LIST (NLST)**

This command causes a directory listing to be sent from server to user site. The pathname should specify a directory or other system-specific file group descriptor; a null argument implies the current directory. The server will return a stream of names of files and no other information. The data will be transferred in ASCII or EBCDIC type over the data connection as valid pathname strings separated by <CRLF> or <NL>. (Again the user must ensure that the TYPE is correct.)

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

#### SITE PARAMETERS (SITE)

This command is used by the server to provide services specific to his system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol. The nature of these services and the specification of their syntax can be stated in a reply to the HELP SITE command.

#### STATUS (STAT)

This command shall cause a status response to be sent over the TELNET connection in the form of a reply. The command may be sent during a file transfer (along with the TELNET IP and Synch signals--see the Section on FTP Commands) in which case the server will respond with the status of the operation in progress, or it may be sent between file transfers. In the latter case the command may have an argument field. If the argument is a pathname, the command is analogous to the "list" command except that data shall be transferred over the TELNET connection. If a partial pathname is given, the server may respond with a list of file names or attributes associated with that specification. If no argument is given, the server should return general status information about the server FTP process. This should include current values of all transfer parameters and the status of connections.

#### HELP (HELP)

This command shall cause the server to send helpful information regarding its implementation status over the TELNET connection to the user. The command may take an argument (e.g., any command name) and return more specific information as a response. The reply is type Oxx, general system status. It is suggested that HELP be allowed before entering a USER command. The server may use this reply to specify site-dependent parameters, e.g., in response to HELP SITE.

#### NOOP (NOOP)

This command does not affect any parameters or previously entered commands. It specifies no action other than that the server send a 200 reply.

### MISCELLANEOUS COMMANDS

There are several functions that utilize the services of file transfer but go beyond it in scope. These are the Mail and Remote Job Entry functions. It is suggested that these become auxiliary protocols that can assume recognition of file transfer commands on the part of the server, i.e., they may depend on the core of FTP commands. The command sets specific to Mail and RJE will be given in separate documents.

Commands that are closely related to file transfer but not proven essential to the protocol may be implemented by servers on an experimental basis. The command name should begin with an X and may be listed in the HELP command. The official command set is expandable from these experiments; all experimental commands or proposals for expanding the official command set should be announced via RFC. An example of a current experimental command is:

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

**Change Working Directory (XCWD)**

This command allows the user to work with a different directory or dataset for file storage or retrieval without altering his login or accounting information. Transfer parameters are similarly unchanged. The argument is a pathname specifying a directory or other system dependent file group designator.

**FTP REPLIES**

The server sends FTP replies over the TELNET connection in response to user FTP commands. The FTP replies constitute the acknowledgment or completion code (including errors). The FTP-server replies are formatted for human or program interpretation. Single line replies consist of a leading three-digit numeric code followed by a space, followed by a one-line text explanation of the code. For replies that contain several lines of text, the first line will have a leading three-digit numeric code followed immediately by the character "-" (Hyphen, ASCII code 45), and possibly some text. All succeeding continuation lines except the last are constrained NOT to begin with three digits; the last line must repeat the numeric code of the first line and be followed immediately by a space. For example:

100-First Line  
Continuation Line  
Another Line  
100 Last Line

It is possible to nest (but not overlap) a reply within a multi-line reply. The same format for matched number-coded first and last lines holds.

The numeric codes are assigned by groups and for ease of interpretation by programs in a manner consistent with other protocols such as the RJE protocol. The three digits of the code are to be interpreted as follows:

1. The first digit specifies type of response as indicated below:

- 0xx These replies are purely informative and constitute neither a positive nor a negative acknowledgment.
- 1xx Informative replies to status inquiries. These constitute a positive acknowledgment to the status command.
- 2xx Positive acknowledgment of previous command or other successful action.
- 3xx Incomplete information. Activity cannot proceed without further specification and input.
- 4xx Unsuccessful reply. The request is correctly specified but the server is unsuccessful in correctly fulfilling it.
- 5xx Incorrect or illegal command. The command or its parameters were invalid or incomplete from a syntactic viewpoint, or the command is inconsistent with a previous command. The command in question has been completely ignored.
- 6xx-9xx Reserved for future expansion.

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

2. The second digit specifies the general category to which the response refers:

- x00-x29 General purpose replies, not assignable to other categories.
- x3x Primary access. Informative replies to the "log-on" attempt.
- x4x Secondary access. The primary server is commenting on its ability to access a secondary service.
- x5x FTP results.
- x6x RJE results.
- x7x Mail Protocol results.
- x8x-x9x Reserved for future expansion.

3. The final digit specifies a particular message type. Since the code is designed for an automaton process to interpret, it is not necessary for every variation of a reply to have a unique number. Only the basic meaning of replies need have unique numbers. The text of a reply can explain the specific reason for that reply to a human user.

Each TELNET line delimited by a numeric code and the TELNET EOL (or group of text lines bounded by coded lines) that is sent by the server is intended to be a complete reply message. It should be noted that the text of replies is intended for a human user. Only the reply codes and in some instances the first line of text are intended for programs.

The assigned reply codes relating to FTP are:

- 000 Announcing FTP.
- 010 Message from system operator.
- 020 Exected delay.
- 030 Server availability information.
- 050 FTP commentary or user information.
- 100 System status reply.
- 110 System busy doing...
- 150 File status reply.
- 151 Directory listing reply.
- 200 Last command received correctly.
- 201 An ABORT has terminated activity, as requested.
- 202 Abort request ignored, no activity in progress.
- 230 User is "logged in". May proceed.
- 231 User is "logged out". Service terminated.
- 232 Logout command noted, will complete when transfer done.
- 233 User is "logged out". Parameters reinitialized.
- 250 FTP file transfer started correctly.
- 251 FTP Restart-marker reply.  
Text is: MARK yyyy = mmmm  
where 'yyyy' is user's data stream marker (yours)  
and mmmm is server's equivalent marker (mine)  
(Note the spaces between the markers and '=').
- 252 FTP transfer completed correctly.
- 253 Rename completed.
- 254 Delete completed.
- 257 Closing the data connection, transfer completed.
- 300 Connection greeting message, awaiting input.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

301 Current command incomplete (no <CRLF> for long time).  
330 Enter password.  
331 Enter account (if account required as part of login sequence).  
332 Login first, please.  
400 This service not implemented.  
401 This service not accepting users now, goodbye.  
402 Command not implemented for requested value or action.  
430 Log-on time or tries exceeded, goodbye.  
431 Log-on unsuccessful. User and/or password invalid.  
432 User not valid for this service.  
433 Cannot transfer files without valid account. Enter account and resend command.  
434 Log-out forced by operator action. Phone site.  
435 Log-out forced by system problem.  
436 Service shutting down, goodbye.  
450 FTP: File not found.  
451 FTP: File access denied to you.  
452 FTP: File transfer incomplete, data connection closed.  
453 FTP: File transfer incomplete, insufficient storage space.  
454 FTP: Cannot connect to your data socket.  
455 FTP: File system error not covered by other reply codes.  
456 FTP: Name duplication; rename failed.  
457 FTP: Transfer parameters in error.  
500 Last command line completely unrecognized.  
501 Syntax of last command is incorrect.  
502 Last command incomplete, parameters missing.  
503 Last command invalid (ignored), illegal parameter combination.  
504 Last command invalid, action not possible at this time.  
505 Last command conflicts illegally with previous command(s).  
506 Last command not implemented by the server.  
507 Catchall error reply.  
550 Bad pathname specification (e.g., syntax error).

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

## DECLARATIVE SPECIFICATIONS

### MINIMUM IMPLEMENTATION

In order to make FTP workable without needless error messages, the following minimum implementation is required for servers:

```
TYPE - ASCII Non-print
MODE - Stream
STRUCTURE - File, Record
BYTE - 8
COMMANDS - USER, BYE, SOCK,
            TYPE, BYTE, MODE, STRU,
            for the default values
            RETR, STOR,
            NOOP.
```

The initial default values for transfer parameters are:

```
TYPE - ASCII Non-print
BYTE - 8
MODE - Stream
STRU - File
```

All Hosts must accept the above as the standard defaults.

### CONNECTIONS

The server protocol interpreter shall "listen" on Socket 3. The user or user protocol interpreter shall initiate the full-duplex TELNET connections performing the ARPANET standard initial connection protocol (ICP) to server Socket 3. Server- and user- processes should follow the conventions of the TELNET protocol as specified in NIC #7104. Servers are under no obligation to provide for editing of command lines and may specify that it be done in the user Host. The TELNET connections shall be closed by the server at the user's request after all transfers and replies are completed.

The user-DTP must "listen" on the specified data sockets (send and/or receive); these may be the default user sockets (U+4) and (U+5) or a socket specified in the SOCK command. The server shall initiate the data connection from his own fixed sockets (S+2) and (S+3) using the specified user data socket and byte size (default - 8 bits). The direction of the transfer and the sockets used will be determined by the FTP service command.

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

When data is to be transferred between two servers, A and B (refer to Figure 2), the user-PI, C, sets up TELNET connections with both server-PI's. He then sends A's fixed sockets, S(A), to B in a SOCK command and B's to A; replies are returned. One of the servers, say A, is then sent a PASV command telling him to "listen" on his data sockets rather than initiate an RFC when he receives a transfer service command. When the user-PI receives an acknowledgment to the PASV command, he may send (in either order) the corresponding service commands to A and B. Server B initiates the RFC and the transfer proceeds. The command-reply sequence is listed below where the messages are vertically synchronous but horizontally asynchronous:

User-PI - Server A	User-PI - Server B
-----	-----
C->A : ICP	C->B : ICP
C->A : SOCK HOST-B, SKT-S(B)	C->B : SOCK HOST-A, SKT-S(A)
A->C : 200 Okay	B->C : 200 Okay
C->A : PASV	
A->C : 200 Okay	C->B : RETR
C->A : STOR	

The data connection shall be closed by the server under the conditions described in the Section on Establishing Data Connections. If the server wishes to close the connection after a transfer where it is not required, he should do so immediately after the file transfer is completed. He should not wait until after a new transfer command is received because the user-process will have already tested the data connection to see if it needs to do a "listen"; (recall that the user must "listen" on a closed data socket BEFORE sending the transfer request). To prevent a race condition here, the server sends a secondary reply (257) after closing the data connection (or if the connection is left open, a "file transfer completed" reply (252) and the user-PI should wait for one of these replies before issuing a new transfer command.

#### COMMANDS

The commands are TELNET character string transmitted over the TELNET connections as described in the Section on FTP Commands. The command functions and semantics are described in the Section on Access Control Commands, Transfer Parameter Commands, FTP Service Commands, and Miscellaneous Commands. The command syntax is specified here.

The commands begin with a command code followed by an argument field. The command codes are four or fewer alphabetic characters. Upper and lower case alphabetic characters are to be treated identically. Thus any of the following may represent the retrieve command:

RETR Retr retr ReTr rETr

This also applies to any symbols representing parameter values, such as A or a for ASCII TYPE. The command codes and the argument fields are separated by one or more spaces.

The argument field consists of a variable length character string ending with the character sequence <CRLF> (Carriage Return, Linefeed) for NVT-ASCII representation; for other negotiated languages a different end of line character might be used. It should be noted that the server is to take NO action until the end of line code is received.

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

The syntax is specified below in NVT-ASCII. All characters in the argument field are ASCII characters including any ASCII represented decimal integers. Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

The following are all the currently defined FTP commands:

```
USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <acctno> <CRLF>
REIN <CRLF>
BYE <CRLF>
BYTE <SP> <byte size> <CRLF>
SOCK <SP> <Host-socket> <CRLF>
PASV <CRLF>
TYPE <SP> <type code> <CRLF>
STRU <SP> <structure code> <CRLF>
MODE <SP> <mode code> <CRLF>
RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal integer> [<SP> R <SP> <decimal integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
DELE <SP> <pathname> <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>
```

The syntax of the above argument fields (using BNF notation where applicable ) is:

```
<username> ::= <string>
<password> ::= <string>
<acctno> ::= <string>
<string> ::= <char>|<char><string>
<char> ::= any of the 128 ASCII characters except <CR> and <LF>
<marker> ::= <pr string>
<pr string> ::= <pr char>|<pr char><pr string>
<pr char> ::= any ASCII code 33. through 126., printable
               characters
<byte size> ::= any decimal integer 1 through 255
<Host-socket> ::= <socket>|<Host number>, <socket>
<Host-number> ::= a decimal integer specifying an ARPANET Host.
<socket> ::= decimal integer between 0 and (2**32)-1
<form code> ::= NTC
<type code> ::= A [<SP> <form code>] E [<SP> <form code>] I
L <SP> <byte size>
<structure code> ::= FR
<mode code> ::= SPC
<pathname> ::= <string>
```

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

**SEQUENCING OF COMMANDS AND REPLIES**

The communication between the user and server is intended to be an alternating dialogue. As such, the user issues an FTP command and the server responds with a prompt primary reply. The user should wait for this initial primary success or failure response before sending further commands.

Certain commands require a second reply for which the user should also wait. These replies may, for example, report on the progress or completion of file transfer or the closing of the data connection. They are secondary replies to file transfer commands.

The third class of replies are informational and spontaneous replies which may arrive at any time. The user-PI should be prepared to receive them. These replies are listed below as spontaneous.

One important group of spontaneous replies is the connection greetings. Under normal circumstances, a server will send a 300 reply, "awaiting input", when the ICP is completed. The user should wait for this greeting message before sending any commands. If the server is unable to accept input right away, he should send a 000 "announcing FTP" or a 020 "expected delay" reply immediately and a 300 reply when ready. The user will then know not to hang up if there is a delay.

The table below lists alternative success and failure replies for each command. These must be strictly adhered to; a server may substitute text in the replies, but the meaning and action implied by the code numbers and by the specific command reply sequence cannot be altered.

**COMMAND-REPLY CORRESPONDENCE TABLE**

COMMAND	SUCCESS	FAILURE
USER	230, 330	430-432, 500-505, 507
PASS	230, 330	430-432, 500-507
ACCT	230	430-432, 500-507
REIN	232, 233	401, 436, 500-507
Secondary Reply	300	
BYE	231, 232	500-505, 507
BYTE	200, 331	402, 500-505, 507
SOCK	200, 331	500-505, 507
PASV	200, 331	500-507
TYPE	200, 331	402, 500-505, 507
STRU	200, 331	500-505, 507
MODE	200, 331	402, 500-505, 507
RETR	250	402, 433, 450, 451, 454, 455, 457, 500-505, 507, 550
Secondary Reply	252, 257	452
STOR	250	402, 433, 451, 454, 455, 457, 500-505, 507, 550
Secondary Reply	252, 257	452, 453
APPE	250	402, 433, 451, 454, 455, 457, 500-507, 550
Secondary Reply	252, 257	452, 453
ALLO	200, 331	402, 500-507
REST	200, 331	500-507
RNFR	200	402, 433, 450, 451, 455, 500-507, 550
RNTO	253	402, 433, 450, 451, 455, 456, 500-507, 550
ABOR	201, 202, 331	500-507

FILE TRANSFER PROTOCOL  
RFC 542 NIC 17759 (Aug. 12, 1973)

DELETE	254	402, 433, 450, 451, 455, 500-507, 550
LIST	250	402, 433, 450, 451, 454, 455, 457, 500-507, 550
NLST	252, 257	452 402, 433, 450, 451, 454, 455, 457, 500-507, 550
Secondary Reply	250	
SITE	252, 257	452 402, 500-507
STAT	200, 331	450, 451, 455, 500-507, 550
	100, 110,	
	150, 151, 331	
HELP	030, 050	500-507
NOOP	200	500-505, 507
Spontaneous Replies	000, 010, 020,	400, 401, 434-436
		300, 301, 251, 255

#### TYPICAL FTP SCENARIOS

TIP User wanting to transfer file from Host X to local printer:

1. TIP user opens TELNET connections by ICP to Host X socket 3.
2. The following commands and replies are exchanged:

TIP	HOST X
<----- 300 Awaiting input <CRLF>	
USER username <CRLF> ----->	
<----- 330 Enter Password <CRLF>	
PASS password <CRLF> ----->	
<----- 230 User logged in <CRLF>	
SOCK 65538 <CRLF> ----->	
<----- 200 Command received OK<CRLF>	
RETR this.file <CRLF> ----->	
(Host X initiates data connection to TIP socket 65538, i.e., PORT 1 receive)	
<----- 250 File transfer started <CRLF>	
<----- 252 File transfer completed <CRLF>	
BYE<CRLF> ----->	
<----- 231 User logged out <CRLF>	

3. Host X closes the TELNET and data connections.

Note: The TIP user should be in line mode.

**FILE TRANSFER PROTOCOL**  
**RFC 542 NIC 17759 (Aug. 12, 1973)**

User at Host U wanting to transfer files to/from Host S:

In general the user will communicate to the server via a mediating user-FTP process. The following may be a typical scenario. The user-FTP prompts are shown in parentheses, '---->' represents commands from Host U to Host S, and '<----' represents replies from Host S to Host U.

LOCAL COMMANDS BY USER	ACTION INVOLVED
ftp (host) multics<CR>	ICP to Host S, socket 3, establishing TELNET connections <---- 330 Awaiting input <CRLF>
username Doe <CR>	USER Doe<CRLF>---->
password mumble <CR>	<---- 330 password<CRLF> PASS mumble<CRLF>---->
retrieve (local type) ASCII<CR> (local pathname) test 1 <CR> (for.pathname) testp11<CR>	<---- 230 Doe logged in.<CRLF> User-FTP opens local file in ASCII. RETR test.p11<CRLF> ----> Server makes data connection to
(U+4)	<---- 250 File transfer starts
<CRLF>	<---- 252 File transfer
complete<CRLF>	TYPE I<CRLF> ---->
type Image<CR>	<---- 200 Command OK<CRLF>
byte 36<CR>	BYTE 36<CR>LF ----> <---- 200 Command OK<CRLF>
store (local type) image<CR> (local pathname) file dump<CR> (for.pathname) >udd>cn>fd<CR>	User-FTP opens local file in Image. STOR >udd>cn>fd<CRLF> ----> <---- 451 Access denied<CRLF>
terminate	BYE <CRLF> ----> Server closes all connections.

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

Jon Postel  
Nancy Neigus  
RFC 640, NIC 30843 (June 5, 1974)

Revised FTP Reply Codes

This document describes a revised set of reply codes for the File Transfer Protocol.

The aim of this revision is to satisfy the goal of using reply codes to enable the command issuing process to easily determine the outcome of each command. The user protocol interpreter should be able to determine the success or failure of a command by examining the first digit of the reply code.

An important change in the sequencing of commands and replies which may not be obvious in the following documents concerns the establishment of the data connection.

In the previous FTP specifications when an actual transfer command (STOR, RETR, APPE, LIST, NLIST, MLFL) was issued the preliminary reply was sent after the data connection was established. This presented a problem for some user protocol interpreters which had difficulty monitoring two connections asynchronously.

The current specification is that the preliminary reply to the actual transfer commands indicates that the file can be transferred and either the connection was previously established or an attempt is about to be made to establish the data connection.

This reply code revision is a modification of the protocol described in RFC 542, that is to say that the protocol implementation associated with socket number 21 (decimal) is the protocol specified by the combination of RFC 542 and this RFC.

A note of thanks to those who contributed to this work: Ken Pogran, Mark Krilanovich, Wayne Hathway, and especially Nancy Neigus.

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

Nancy Neigus  
Ken Pogran  
Jon Postel  
RFC 640, NIC 30843 (June 5, 1974)

### A New Schema for FTP Reply Codes

Replies to File Transfer Protocol commands were devised to ensure the synchronization of requests and actions in the process of file transfer, and to guarantee that the user process always knows the state of the Server. Every command must generate at least one reply, although there may be more than one; in the latter case, the multiple replies must be easily distinguished. In addition, some commands occur in sequential groups, such as USER, PASS and ACCT, or RNFR and RNTO. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning.

Details of the command-reply sequence will be made explicit in a state diagram.

An FTP 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 intended for the human user. It is intended that the three digits contain enough encoded information that the user-process (the User-PI described in RFC 542) will not need to 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, so there are likely to be varying texts for each reply code.

Formally, a reply is defined to contain the 3-digit code, followed by Space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the TELNET end-of-line code. There will be cases, however, where the text is longer than a single line. In these cases the complete text must be bracketed so the User-process knows when it may stop reading the reply (i.e. stop processing input on the TELNET connection) and go do other things. This requires a special format on the first line to indicate that more than one line is coming, and another on the last line to designate it as the last. At least one of these must contain the appropriate reply code to indicate the state of the transaction. To satisfy all factions it was decided that both the first and last line codes should be the same.

Thus the format for multi-line replies is that the first line will begin with the exact required reply code, followed immediately by a

Hyphen, "-" (also known as Minus), followed by text. The last line will begin with the same code, followed immediately by Space <SP>, optionally some text, and TELNET <eol>.

For example:

```
123-First line
Second line
 234 A line beginning with numbers
123 The last line
```

The user-process then simply needs to search for the second occurrence of the same reply code, followed by <SP> (Space), at the beginning of a line, and ignore all intermediary lines. If an intermediary line begins with a 3-digit number, the Server must pad the front to avoid confusion.

This scheme allows standard system routines to be used for reply information (such as for the STAT reply), with "artificial" first and last lines tacked on. In the rare cases where these routines are able to generate three digits and a Space at the beginning of any line, the beginning of each text line should be offset by some neutral text, like Space.

This scheme assumes that multi-line replies may not be nested. We have found that, in general, nesting of replies will not occur, except for random system messages (called spontaneous replies in the previous FTP incarnations) which may interrupt another reply. Spontaneous replies are no longer defined; system messages (i.e. those not processed by the FTP server) will NOT carry reply codes and may occur anywhere in the command-reply sequence. They may be ignored by the User-process as they are only information for the human user.

The three digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated response by the user-process. The first digit denotes whether the response is good, bad or incomplete. (Referring to the state diagram) an unsophisticated user-process will be able to determine its next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A user-process that wants to know approximately what kind of error occurred (e.g. file system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information (e.g. RNTO command without a preceding RNFR.).

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

There are four values for the first digit of the reply code:

1yz Positive Preliminary reply

The requested action is being initiated; expect another reply before proceeding with a new command. (The user-process sending another command before the completion reply would be in violation of protocol; but server-FTP processes should queue any commands that arrive while a preceding command is in progress.) This type of reply can be used to indicate that the command was accepted and the user-process may now pay attention to the data connections, for implementations where simultaneous monitoring is difficult.

2yz Positive Completion reply

The requested action has been successfully completed. A new request may be initiated.

3yz Positive Intermediate reply

The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The user should send another command specifying this information. This reply is used in command sequence groups.

4yz Transient Negative Completion reply

The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again. The user should return to the beginning of the command sequence, if any. It is difficult to assign a meaning to "transient", particularly when two distinct sites (Server and User-processes) have to agree on the interpretation. Each reply in the 4yz category might have a slightly different time value, but the intent is that the user-process is encouraged to try again. A rule of thumb in determining if a reply fits into the 4yz or the 5yz (Permanent Negative) category is that replies are 4yz if the commands can be repeated without any change in command form or in properties of the User or Server (e.g. the command is spelled the same with the same arguments used; the user does not change his file access or user name; the server does not put up a new implementation.)

5yz Permanent Negative Completion reply

The command was not accepted and the requested action did not take place. The User-process is discouraged from repeating the exact

request (in the same sequence). Even some "permanent" error conditions can be corrected, so the human user may want to direct his User-process to reinitiate the command sequence by direct action at some point in the future (e.g. after the spelling has been changed, or the user has altered his directory status.)

The following function groupings are encoded in the second digit:

- x0z Syntax - These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, unimplemented or superfluous commands.
- x1z Information - These are replies to requests for information, such as status or help.
- x2z Connections - Replies referring to the TELNET and data connections.
- x3z Authentication and accounting - Replies for the logon process and accounting procedures.
- x4z Unspecified as yet
- x5z File system - These replies indicate the status of the Server file system vis-a-vis the requested transfer or other file system action.

The third digit gives a finer gradation of meaning in each of the function categories, specified by the second digit. The list of replies below will illustrate this. Note that the text associated with each reply is suggestive, rather than mandatory, and may even change according to the command with which it is associated. The reply codes, on the other hand, should strictly follow the specifications in the last section; that is, Server implementations should not invent new codes for situations that are only slightly different from the ones described here, but rather should adapt codes already defined. If additional codes are found to be necessary, the details should be submitted to the FTP committee, through Jon Postel.

A command such as TYPE or ALLO whose successful execution does not offer the user-process any new information will cause a 200 reply to be returned. If the command is not implemented by a particular Server-FTP process because it has no relevance to that computer system, for example ALLO at a TENEX site, a Positive Completion reply is still desired so that the simple User-process knows it can proceed with its course of action. A 202 reply is used in this case with, for example, the reply text: "No storage

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

allocation necessary." If, on the other hand, the command requests a non-site-specific action and is unimplemented, the response is 502. A refinement of that is the 504 reply for a command that IS implemented, but that requests an unimplemented parameter.

200 Command okay  
500 Syntax error, command unrecognized  
[This may include errors such as command line too long.]  
501 Syntax error in parameters or arguments  
202 Command not implemented, superfluous at this site.  
502 Command not implemented  
503 Bad sequence of commands  
504 Command not implemented for that parameter

110 Restart marker reply.  
In this case the text is exact and not left to the particular implementation; it must read:  
    MARK yyyy = mmmm  
where yyyy is User-process data stream marker, and mmmm is Server's equivalent marker. (note the spaces between the markers and "=".)  
211 System status, or system help reply  
212 Directory status  
213 File status  
214 Help message (on how to use the server or the meaning of a particular non-standard command. This reply is useful only to the human user.)

120 Service ready in nnn minutes  
220 Service ready for new user  
221 Service closing TELNET connection (logged off if appropriate)  
421 Service not available, closing TELNET connection.  
[This may be a reply to any command if the service knows it must shut down.]  
125 Data connection already open; transfer starting  
225 Data connection open; no transfer in progress  
425 Can't open data connection  
226 Closing data connection; requested file action successful (for example, file transfer or file abort.)  
426 Connection trouble, closed; transfer aborted.  
227 Entering [passive, active] mode

230 User logged on, proceed

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

530 Not logged in  
331 User name okay, need password  
332 Need account for login  
532 Need account for storing files

150 File status okay; about to open data connection.  
250 Requested file action okay, completed.  
350 Requested file action pending further information  
450 Requested file action not taken: file unavailable (e.g.  
      file not found, no access)  
550 Requested action not taken: file unavailable (e.g. file  
      busy)  
451 Requested action aborted: local error in processing  
452 Requested action not taken: insufficient storage space  
      in system  
552 Requested file action aborted: exceeded storage  
      allocation (for current directory or dataset)  
553 Requested action not taken: file name not allowed  
354 Start mail input; end with <CR><LF>.<CR><LF>

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

Command-Reply Sequences

In this section, the command-reply sequence is presented. Each command is listed with its possible replies; command groups are listed together. Preliminary replies are listed first (with their succeeding replies under them), then positive and negative completion, and finally intermediary replies with the remaining commands from the sequence following. This listing forms the basis for the state diagrams, which will be presented separately.

ICP  
120  
220  
220  
421

Logon

USER  
230  
530  
500, 501, 421  
331, 332  
PASS  
230  
202  
530  
500, 501, 503, 421  
332  
ACCT  
230  
202  
530  
500, 501, 503, 421

Logoff

QUIT  
221  
500  
REIN  
120  
220

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

220  
421  
500, 502

**Transfer parameters**

SOCK  
    200  
    500, 501, 421, 530  
PASV  
    227  
    500, 501, 502, 421, 530  
ACTV  
    227  
    202  
    500, 501, 421, 530  
BYTE, MODE, TYPE, STRU  
    200  
    500, 501, 504, 421, 530

**File action commands**

ALLO  
    200  
    202  
    500, 501, 504, 421, 530  
REST  
    500, 501, 502, 421, 530  
    350  
STOR  
    125, 150  
        (110)  
    226, 250  
        425, 426, 451, 552  
    532, 450, 452, 553  
    500, 501, 421, 530  
RETR  
    125, 150  
        (110)  
    226, 250  
        425, 426, 451  
    450, 550  
    500, 501, 421, 530

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

LIST, NLST  
125, 150  
226, 250  
425, 426, 451  
450  
500, 501, 502, 421, 530

APPE  
125, 150  
(110)  
226, 250  
425, 426, 451, 552  
532, 450, 550, 452, 553  
500, 501, 502, 421, 530

MLFL  
125, 150  
226, 250  
425, 426, 451, 552  
532, 450, 550, 452, 553  
500, 501, 502, 421, 530

RNFR  
450, 550  
500, 501, 502, 421, 530  
350

RNTO  
250  
532, 553  
500, 501, 502, 503, 421, 530

DELETE  
250  
450, 550  
500, 501, 502, 421, 530

ABOR  
225, 226  
500, 501, 502, 421

MAIL  
354  
250  
451, 552  
450, 550, 452, 553  
500, 501, 502, 421, 530

Informational commands

STAT  
211, 212, 213  
450  
500, 501, 502, 421, 530

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

HELP  
211, 214  
500, 501, 502, 421

Miscellaneous commands

SITE  
200  
202  
500, 501, 530  
NOOP  
200  
500

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

Jon Postel  
March 24, 1976

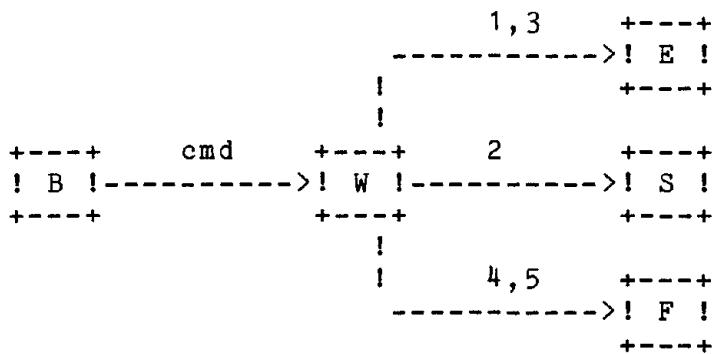
FTP State Diagrams

Here we present state diagrams for a very simple minded FTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of FTP commands or command sequences.

The command groupings were determined by constructing a model for each command then collecting together the commands with structurally identical models.

For each command or command sequence there are three possible outcomes: success (S), failure (F), and error (E). In the state diagrams below we use the symbol B for "begin", and the symbol W for "wait for reply".

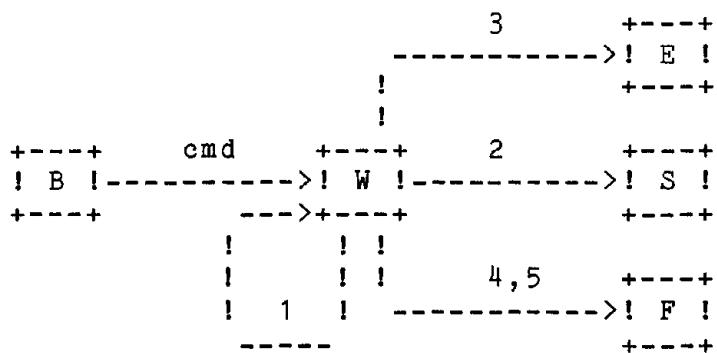
We first present the diagram that represents the largest group of FTP commands:



This diagram models the commands:

ABOR, ACTV, ALLO, BYTE, DELE, HELP, MODE, NOOP, PASV, QUIT, SITE,  
SOCK, STAT, STRU, TYPE.

The other large group of commands is represented by a very similar diagram:



This diagram models the commands:

APPE, (ICP), LIST, MLFL, NLST, REIN, RETR, STOR.

Note that this second model could also be used to represent the first group of commands, the only difference being that in the first group the 100 series replies are unexpected and therefore treated as error, while the second group expects (some may require) 100 series replies.

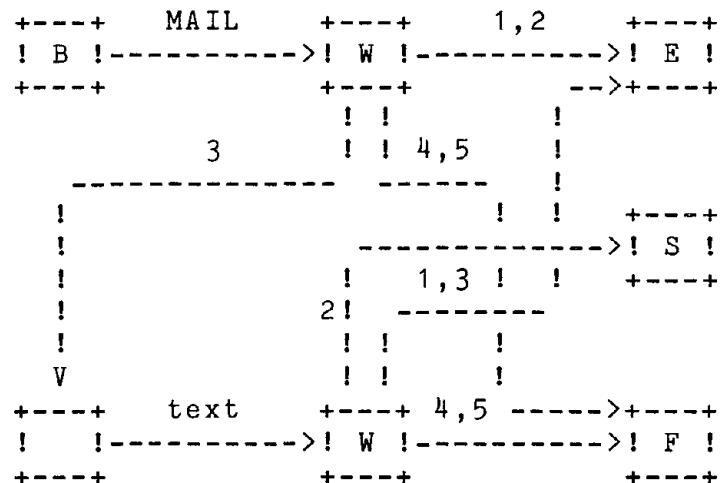
The remaining diagrams model command sequences, perhaps the simplest of these is the rename sequence:

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

+---+	RNFR	+---+	1,2	+---+
! B !----->!	W !----->!	E !		
+---+	+---+	-->+---+		
	!!	!		
	3	!! 4,5	!	
	-----	-----	!	
!		!!	! +---+	
!		----->!	S !	
!		!! 1,3 !	! +---+	
!	2!	-----		
!		!!	!	
V		!!	!	
+---+	RNTO	+---+	4,5 ----->+---+	
! !----->!	W !----->!	F !		
+---+	+---+	+---+		

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

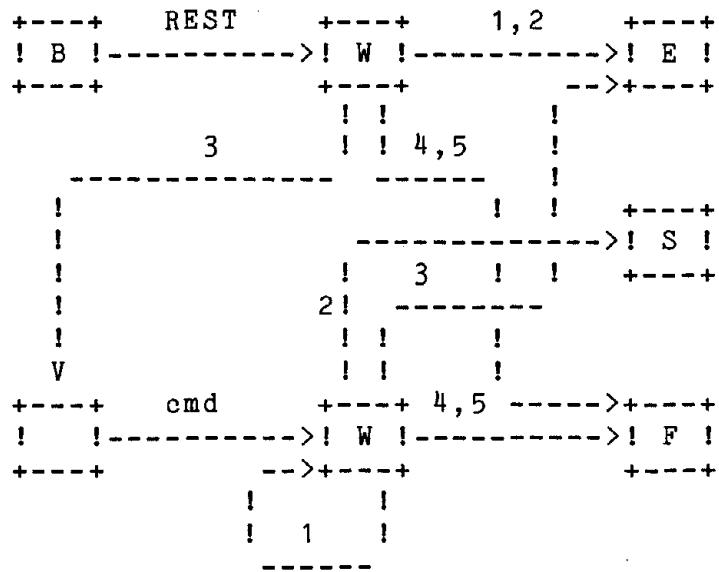
A very similar diagram models the Mail command:



Note that the "text" here is a series of lines sent from the user to the server with no response expected until the last line is sent, recall that the last line must consist only of a single period.

Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

The next diagram is a simple model of the Restart command:

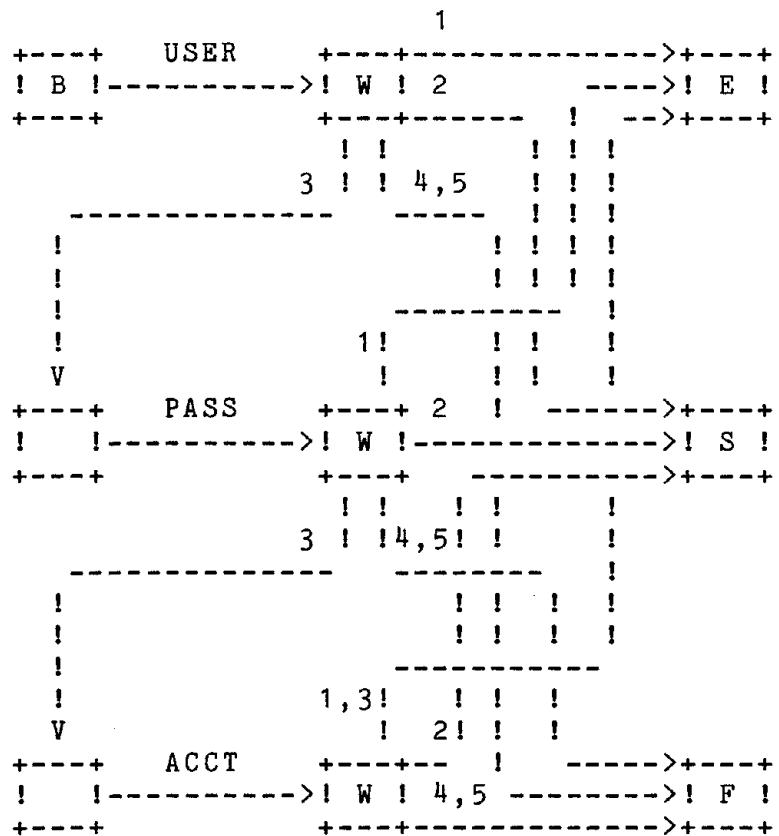


Where "cmd" is APPE, STOR, RETR, or MLFL.

We note that the above three models are similar, in fact the Mail diagram and the Rename diagram are structurally identical. The Restart differs from the other two only in the treatment of 100 series replies at the second stage.

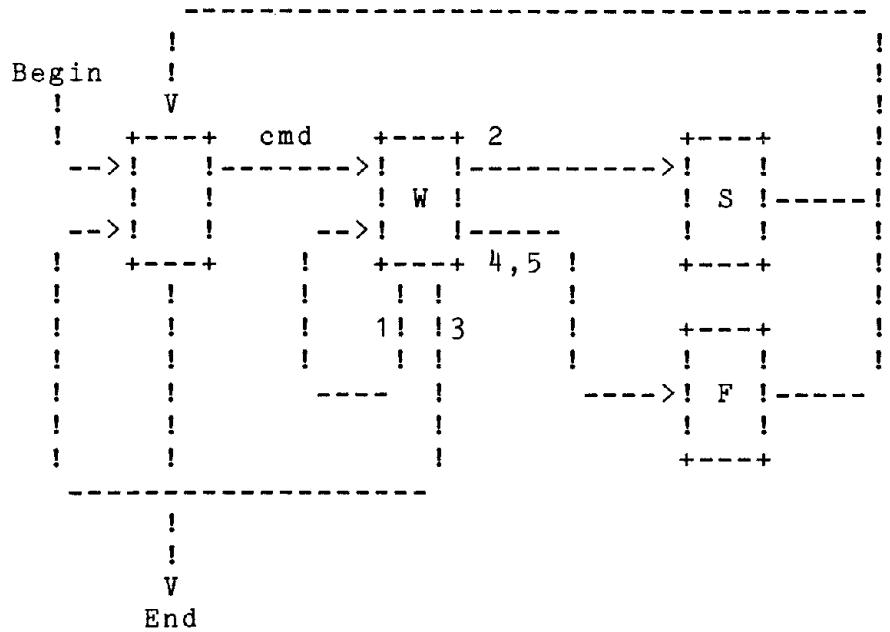
Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

The most complicated diagram is for the Logon sequence:



Revised FTP Reply Codes  
RFC 640, NIC 30843 (June 5, 1974)

Finally we present a generalized diagram that could be used to model the command and reply interchange:



K. Harrenstien (SRI-KL)  
RFC 737, NIC 42217 (Oct. 31, 1977)

FTP Extension: XSEN

This note describes an extension to the File Transfer Protocol which provides for "sending" a message to a logged-in user, as well as variants for mailing it normally whether the user is logged in or not.

Several systems have a SEND command or program which sends a message directly to a user's terminal. On the SAIL (SU-AI) and ITS (MIT-(AI/ML/MC/DMS)) systems the concept has been broadened to allow SENDing to users on other network sites; to support this, three new FTP commands were added which have a syntax identical to the existing MAIL command. For reference, the latter is:

MAIL <SP> <recipient name> <CRLF>

If accepted, returns 350 reply and considers all succeeding lines to be the message text, terminated by a line containing only a period, upon which a 256 completion reply is returned. Various errors are possible.

The new commands, with their special replies, are:

XSEN -- Send to terminal.

Returns 453 failure reply if the addressee is refusing or not logged in.

XSEM -- Send, Mail if can't.

Returns 009 notification reply if message cannot be SENT.

XMAS -- Mail And Send. (couldn't resist this one)

No special replies.

NOTE: for XSEM and XMAS, it is the mailing which determines success, not the SENDing, although XSEM as implemented uses a 009 reply (in addition to the normal success/failure code) to indicate that because the SEND failed, an attempt is being made to mail the message instead. There are no corresponding variants for MLFL, since messages transmitted in this way are generally short, and neither I nor Brian Harvey (implementing respectively the ITS and SAIL servers) wanted to bother.



TENEX FTP EXTENSIONS FOR PAGED FILES  
RFC 683, NIC 32251 (Apr. 8, 1975)

R. Clements (BBN)  
RFC 683, NIC 32251 (Apr. 8, 1975)

TENEX FTP EXTENSIONS FOR PAGED FILES

### Introduction

In response to a long-known need for the ability to transfer TENEX paged files over the net via FTP, the TENEX FTP implementation has been extended.

This implementation is an extension to the "OLD" protocol (RFC 354). It was built after useful discussions with Postel, Neigus, et al. I do not mean to imply that they agreed that this implementation is correct, nor for that matter do I feel it is correct. A "correct" implementation will be negotiated and implemented in the "NEW" protocol (RFC 542), if funding ever appears for that task.

### The Problem(s)

This extension attacks two separate problems: Network reliability and TENEX disk file format's incompatibility with FTP. A checksummed and block-sequence-numbered transmission mode is seriously needed, in my opinion. This mode should also allow data compression.

It is also necessary to handle paged, holey TENEX files. This latter problem, seriously needed for NLS, is the motivation for the current extension.

The former problem requires a new MODE command, if done correctly; probably two MODEs, to allow data compression in addition to checksumming. Actually, I think that is the tip of an iceberg which grows as  $2^{**}N$  for additional sorts of modes, so maybe some mode combination system needs to be dreamed up. Cf the AN, AT, AC, EN, ET, EC TYPES. Also, one should be able to use MODE B and MODE C together (NEW protocol) to gain both the compression and restart facilities if one wanted.

The second problem, TENEX files, are probably a new kind of STRUCTure. However, it should be possible to send a paper tape to a disk file, or vice versa, with the transfer looking like a paged file; so perhaps we are dealing with a data representation TYPE. This argument is a bit strained, though, so a paged STRUCTure is quite likely correct. I admit to feeling very unsure about what is a MODE, what is a TYPE and what is a STRUCTure.

---

Preceding page blank

TENEX FTP EXTENSIONS FOR PAGED FILES  
RFC 683, NIC 32251 (Apr. 8, 1975)

The (Incorrect) choices made

Having decided that new MODEs and STRUCTures were needed, I instead implemented the whole thing as a single new TYPE. After all, I rationalize, checksumming the data on the network (MODE) and representing the data in the processing system as a checksummed TYPE are really just a matter of where you draw the imaginary line between the net and the data. Also, a single new TYPE command reduced the size of the surgery required on the FTP user and server programs.

Implementation details

The name of the new TYPE is "XTP". I propose this as a standard for all the Key Letter class of FTP commands: the "X" stands for "experimental" -- agreed on between cooperating sites. The letter after the "X" is signed out from the protocol deity by an implementor for a given system. In this case, "T" is for TENEX. Subsequent letter(s) distinguish among possibly multiple private values of the FTP command. Here "P" is "Paged" type.

TYPE XTP is only implemented for STRU F, BYTE 36, and MODE S.

Information of TYPE XTP is transferred in chunks (I intentionally avoid the words RECORD and BLOCK) which consist of a header and some data. The data in a chunk may be part of the data portion of the file being transferred, or it may be the FDB (File Descriptor Block) associated with the file.

Diversion: the TENEX Disk File

For those not familiar with the TENEX file system, a brief dissertation is included here to make the rest of the implementation meaningful.

A TENEX disk file consists of four things: a pathname, a page table, a (possibly empty) set of pages, and a set of attributes.

The pathname is specified in the RETR or STOR verb. It includes the directory name, file name, file name extension, and version number.

The page table contains up to  $2^{**}18$  entries. Each entry may be EMPTY, or may point to a page. If it is not empty, there are also some page-specific access bits; not all pages of a file need have the same access protection.

A page is a contiguous set of 512 words of 36 bits each.

TENEX FTP EXTENSIONS FOR PAGED FILES  
RFC 683, NIC 32251 (Apr. 8, 1975)

The attributes of the file, in the FDB, contain such things as creation time, write time, read time, writer's byte-size, end of file pointer, count of reads and writes, backup system tape numbers, etc.

NOTE: there is NO requirement that pages in the page table be contiguous. There may be empty page table slots between occupied ones. Also, the end of file pointer is simply a number. There is no requirement that it in fact point at the "last" datum in the file. Ordinary sequential I/O calls in TENEX will cause the end of file pointer to be left after the last datum written, but other operations may cause it not to be so, if a particular programming system so requires.

In fact both of these special cases, "holey" files and end-of-file pointers not at the end of the file, occur with NLS data files. These files were the motivation for the new TYPE.

Meanwhile, back at the implementation,...

Each chunk of information has a header. The first byte, which is the first word (since TYPE XTP is only implemented for BYTE 36) of the chunk, is a small number, currently 6, which is the number of following words which are still in the header. Next come those six words, and then come some data words.

The six header words are:

Word 1: a checksum.

This is a one's complement sum (magnitude and end-around carry) of the six header words and the following data words (but not the leading "6" itself). The sum of all words including the checksum must come out + or - zero.

Word 2: A sequence number.

The first chunk is number 1, the second is number 2, etc.

Word 3: NDW,

the number of data words in this chunk, following the header. Thus the total length of the chunk is 1 (the word containing NHEAD) + NHEAD + NDW. The checksum checks all but the first of these.

TENEX FTP EXTENSIONS FOR PAGED FILES  
RFC 683, NIC 32251 (Apr. 8, 1975)

Word 4: Page number.

If the data is a disk file page, this is the number of that page in the file's page map. Empty pages (holes) in the file are simply not sent. Note that a hole is NOT the same as a page of zeroes.

Word 5: ACCESS.

The access bits associated with the page in the file's page map. (This full word quantity is put into AC2 of an SPACS by the program reading from net to disk.)

Word 6: TYPE.

A code for what type of chunk this is. Currently, only type zero for a data page, and type -3 for an FDB are sent.

After the header are NDW data words. NDW is currently either 1000 octal for a data page or 25 octal for an FDB. Trailing zeroes in a disk file page will soon be discarded, making NDW less than 1000 in that case. The receiving portions of FTP server and user will accept these shortened pages. The sender doesn't happen to send them that way yet.

Verification is performed such that an error is reported if either:

The checksum fails,

The sequence number is not correct,

NDW is unreasonable for the given chunk type, or

The network file ends at some point other than immediately following the data portion of an FDB chunk.

Closing comments

This FTP server and user are in operation on all the BBN systems and at some other sites -- the user being more widely distributed since fewer sites have made local modifications to the user process.

I believe the issues of checksumming and sequencing should be addressed for the "NEW" protocol. I hope the dissertation on TENEX files has been useful to users of other systems. It may explain my lack of comprehension of the "record" concept, for example. A TENEX file is just a bunch of words pointed to by a page table. If those

TENEX FTP EXTENSIONS FOR PAGED FILES  
RFC 683, NIC 32251 (Apr. 8, 1975)

words contain CRLF's, fine -- but that doesn't mean "record" to TENEX. I think this RFC also points out clearly that net data transfers are implemented like the layers of an onion: some characters are packaged into a line. Some lines are packaged into a file. The file is broken into other manageable units for transmission. Those units have compression applied to them. The units may be flagged by restart markers (has anyone actually done that?). The compressed units may be checksummed, sequence numbered, date-and-time stamped, and flagged special delivery. On the other end, the process is reversed. Perhaps MODE, TYPE, and STRU don't really adequately describe the situation. This RFC was written to allow implementors to interface with the new FTP server at TENEX sites which install it. It is also really a request for comments on some of these other issues.



STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

J. Postel  
RFC 678, NIC 31524 (Dec. 19, 1974)

STANDARD FILE FORMATS

Introduction

In an attempt to provide online documents to the network community we have had many problems with the physical format of the final documents. Much of this difficulty lies in the fact that we do not have control or even knowledge of all the processing steps or devices that act on the document file. A large part of the difficulty in the past has been due to some assumptions we made about the rest of the world being approximately like our own environment. We now see that the problems are due to differing assumptions and treatment of files to be printed as documents. We therefore propose to define certain standard formats for files and describe the expected final form for printed copies of such files.

These standard formats are not additional File Transfer Protocol data types/modes/structures, but rather usage descriptions between the originator and ultimate receiver of the file. It may be useful or even necessary at some hosts to construct programs that convert files between common local formats and the standard formats specified here.

The intent is that the author of a document may prepare his/her text and store it in an online file, then advertise that file by name and format (as specified here), such that interested individuals may copy and print the file with full understanding of the characteristics of the format controls and the logical page size.

Standardization Elements

The elements or aspects of a file to be standardized are the character or code set used, the format control procedures, the area of the page to be used for text, and the method to describe overstruck or underlined characters.

The area of the page to be used for text can be confusing to discuss, in an attempt to be clear we define a physical page and a logical page. Please note that the main emphasis of this note is to describe the standard formats in terms of the logical page, and that it is up to each site to map the logical page onto the physical page of each of their devices.

Preceding page blank

STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

Physical Page

The physical page is the medium that carries the text, the height and width of its area are measured in inches.

The typical physical page is a piece of paper eleven inches high and eight and one half inches wide.

Typical print density is 10 characters per inch horizontally and 6 characters per inch vertically. This results in the typical physical page having a maximum capacity of 66 lines and 85 characters per line. It is often the case that printing devices limit the area of the physical page by enforcing margins.

Logical Page

The logical page is the area that can contain text, the height of this area is measured in lines and the width is measured in characters.

A typical logical page is 60 lines high and 72 characters wide.

Code Set

The character encoding will be the network standard Network Virtual Terminal (NVT) code as used in Telnet and File Transfer protocols, that is ASCII in an eight bit byte with the high order bit zero.

Format Control

The format will be controlled by the ASCII format effectors:

Form Feed <FF>

Moves the printer to the top of the next logical page keeping the same horizontal position.

Carriage Return <CR>

Moves the printer to the left edge of the logical page remaining on current line.

STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

Line Feed <LF>

Moves the printer to the next print line, keeping the same horizontal position.

Horizontal Tab <HT>

Moves the printer to the next horizontal tab stop.

The conventional stops for horizontal tabs are every eight characters, that is character positions 9, 17, 25, ... within the logical page.

Note that it is difficult to enforce these conventions and it is therefore recommended that horizontal tabs not be used in document files.

Vertical Tab <VT>

Moves the printer to the next vertical tab stop.

The conventional stops for vertical tabs are every eight lines starting at the first printing line on each logical page, that is lines 1, 9, 17, ... within the logical page.

NOTE: it is difficult to enforce these conventions and it is therefore recommended that vertical tabs not be used in document files.

Back Space <BS>

Moves the printer one character position toward the left edge of the logical page.

Not all these effectors will be used in all format standards, any effectors which are not used in a format standard are ignored.

Page Length

The logical page length will be specified in terms of a number of lines of text.

STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

Page Width

The logical page width will be specified as a number of characters.

Overstriking

Overstriking (note that underlining is a subset of overstriking) may be specified to be done in one or both of the following ways, or not at all:

By Line

The composite line is made up of text segments each terminated by the sequence <CR><NUL> except that the final segment is terminated by the sequence <CR><LF>.

By Character

Each character to be overstruck is to be immediately followed by a <BS> and the overstrike character.

End of Line

The end of line convention is the Telnet end of line convention which is the sequence <CR><LF>. It is recommended that use of <CR> and <LF> be avoided in other than the end of line context.

Standard Formats

Format 1 [Basic Document]

This format is designed to be used for documents to be printed on line printers, which normally have 66 lines to a physical page, but often have forced top and bottom margins of 3 lines each.

Active Format Effectors

<FF>, <CR>, <LF>.

Page Length

60 lines.

Page Width

72 Characters.

Overstriking

By Line.

STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

Format 2 [Terminal]

This format is designed to be used with hard copy terminals, which in the normal case have 66 lines to a physical page. It is expected that there are no top or bottom margins enforced by the terminal or its local system, thus any margins around the physical page break must come from the file.

Active Format Effectors  
    <FF>, <CR>, <LF>, <HT>, <VT>, <BS>.  
Page Length  
    66 lines.  
Page Width  
    72 Characters.  
Overstriking  
    By Character.

Format 3 [Line Printer]

This format is designed to be used with full width (11 by 14 inch paper) line printer output.

Active Format Effectors  
    <FF>, <CR>, <LF>.  
Page Length  
    60 lines.  
Page Width  
    132 Characters.  
Overstriking  
    None.

Format 4 [Card Image]

This format is designed to be used for simulated card input. The page width is 80 characters, each card image is followed by <CR><LF>, thus each card is represented by between 2 and 82 characters in the file. Note that the trailing spaces of a card image need not be present in the file, and that the early occurrence of the <CR><LF> sequence indicates that the remainder of the card image is to contain space characters.

Active Format Effectors  
    <CR>, <LF>.  
Page Length  
    Infinite.  
Page Width  
    80 Characters.

STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

Overstriking

None.

Format 5 [Center Document]

This format is intended for use with documents to be printed on line printers which normally have 66 lines to the physical page but enforce top and bottom margins of 3 lines each. The text is expected to be centered on the paper. If the horizontal printing density is 10 characters per inch and the paper is 8 and 1/2 inches wide then there will be a one inch margin on each side.

Active Format Effectors

<FF>, <CR>, <LF>.

Page Length

60 Lines.

Page Width

65 Characters.

Overstriking

By Line.

Format 6 [Bound Document]

This format is intended for use with documents to be printed on line printers which normally have 66 lines to the physical page but enforce top and bottom margins of 3 lines each. If the horizontal printing density is 10 characters per inch and the paper is 8 and 1/2 inches wide then the text should be positioned such that there is a 1 and 1/2 inch left margin and a one inch right margin.

Active Format Effectors

<FF>, <CR>, <LF>.

Page Length

60 Lines.

Page Width

60 Characters.

Overstriking

By Line.

Implementation Suggestions

Overflow

Overflow can result from two causes, first if the physical page is smaller than the logical page, and second if the actual text in the file violates the standard under which it is being processed.

STANDARD FILE FORMATS  
RFC 678, NIC 31524 (Dec. 19, 1974)

In either case the following suggestions are made to implementors of programs which process files in these formats.

Length

If more lines are processed than fit within the minimum of the physical page and the logical page length since the last <FF>, then the <FF> action should be forced.

Width

If more character positions are processed than fit on the minimum of the physical page width and the logical page width since the last <CR>, then characters are discarded up to the next <CR>.

or

If more character positions are processed than fit on the minimum of the physical page width and the logical page width since the last <CR>, then the <CR> and <LF> actions should be forced.

References

A. McKenzie "TELNET Protocol Specification," Aug. 1973 (NIC 18639).

"USA Standard Code for Information Interchange," United States of America Standards Institute, 1968 (NIC 11246).



MAIL PROTOCOL  
NIC 29588 (Nov. 22, 1977)

Jon Postel, SRI-ARC  
NIC 29588 (Feb. 18, 1976)

MAIL PROTOCOL

Introduction

This document describes the existing mail transmission protocols. The mail transmission protocol is a subset of the File Transfer protocol, consisting of two additional commands to the set of commands described in the specification of the File Transfer protocol.

Old FTP

A. McKenzie "File Transfer Protocol," RFC 454, NIC 14333,  
16-Feb-73.

New FTP

N. Neigus "File Transfer Protocol," RFC 542, NIC 17759, 12-Jul-73.

J. Postel "Revised FTP Reply Codes," RFC 640, NIC 30843, 5-Jun-74.

Commands

Mail File (MLFL)

The intent of this command is to enable a user site to mail data (in the form of a file) to another user at the server site. It should be noted that the files to be mailed are transmitted via the data connection in ASCII or EBCDIC type. (It is the user's responsibility to ensure that the type is correct.) These files should be inserted into the destination user's mailbox by the server in accordance with serving Host mail conventions. The mail may be marked as sent from the particular using HOST and the user specified by the 'USER' command. The argument field may contain one or more Host system or NIC idents (it is recommended that multiple idents be allowed so the same mail can easily be sent to several users), or it may be empty. If the argument field is empty or blank (one or more spaces), then the mail is destined for a printer or other designated place for site mail.

A NIC ident refers to the standard identification described in the Arpanet Directory. A serving host may keep a table mapping NIC idents into system idents, although NIC idents are not required in the implementation. A system ident is the user's normal identification at the serving HOST. The use of system idents

Preceding page blank

MAIL PROTOCOL  
NIC 29588 (Nov. 22, 1977)

would allow a network user to send mail to other users who do not have NIC identification but whose system ident is known.

### Mail (MAIL)

This command allows a user to send mail that is NOT in a file over the TELNET connection. The argument field may contain one or more system or NIC idents, or it may be empty. The idents are defined as above for the MLFL command. After the 'Mail' command is received, the server is to treat the following lines as text of the mail sent by the user. The mail text is to be terminated by a line containing only a single period, that is, the character sequence ".CRLF" in a new line. It is suggested that a modest volume of mail service should be free; i.e., it may be entered before a USER command.

### Reply Codes

The MAIL and MLFL commands have the same reply codes as the Append (APPE) command, with the addition of the reply code for MAIL stating that mail is expected over the Telnet connection.

### Old FTP

350 - Enter mail, terminate with <CR><LF>.<CR><LF>

### New FTP

354 - Start mail input, end with <CR><LF>.<CR><LF>

### Syntax

For consistency in the handling of mail at the various hosts, it is required that all mail sending subsystems or programs use the standard syntax convention documented in RFC 733 (NIC 41952) for the text of the mail. This is essential to enable users, and especially programs, to intelligently process mail.

NOTE: See also RFC 743, which proposes a more efficient scheme for mailing to several recipients.

RFC # 733  
NIC # 41952

Obsoletes: RFC #561 (NIC #18516)  
RFC #680 (NIC #32116)  
RFC #724 (NIC #37435)

STANDARD FOR THE FORMAT OF  
ARPA NETWORK TEXT MESSAGES(1)

21 November 1977

by

David H. Crocker  
The Rand Corporation

John J. Vittal  
Bolt Beranek and Newman Inc.

Kenneth T. Pogran  
Massachusetts Institute of Technology

D. Austin Henderson, Jr. (2)  
Bolt Beranek and Newman Inc.

===== (1) This work was supported by the Defense Advanced Research Projects Agency of the Department of Defense, under contract Nos. N00014-75-C-0661, MDA903-76-C-0212, and DAHC15-73-C0181.

(2) The authors' postal addresses are: D. Crocker, The Rand Corporation, Information Sciences Dept., 1700 Main St., Santa Monica, California 90406; J. Vittal & D. A. Henderson, Bolt Beranek & Newman, 50 Moulton St., Cambridge, Massachusetts 02138; and K. Pogran, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, Massachusetts 02139. The authors' ARPANET addresses are: DCrocker at Rand-Unix, Vittal at BBN-TenexD, Pogran at MIT-Multics, and Henderson at BBN-TenexD.



## PREFACE

ARPA's Committee on Computer-Aided Human Communication (CAHCOM) wishes to promulgate a standard for the format of ARPA Network text message (mail) headers which will reasonably meet the needs of the various message service subsystems on the Network today. The authors of this document constitute the CAHCOM subcommittee charged with the task of developing this new standard.

Essentially, we specify a revision to ARPANET Request for Comments (RFC) 561, "Standardizing Network Mail Headers", and RFC 680, "Message Transmission Protocol". This revision removes and compacts portions of the previous syntax and adds several features to network address specification. In particular, we focus on people and not mailboxes as recipients and allow reference to stored address lists. We expect this syntax to provide sufficient capabilities to meet most users' immediate needs and, therefore, give developers enough breathing room to produce a new mail transmission protocol "properly". We believe that there is enough of a consensus in the Network community in favor of such a standard syntax to make possible its adoption at this time. An earlier draft of this specification was published as RFC #724, "Proposed Official Standard for the Format of ARPA Network Messages" and contained extensive discussion of the background and issues in ARPANET mail standards.

This specification was developed over the course of one year, using the ARPANET mail environment, itself, to provide an on-going forum for discussing the capabilities to be included. More than twenty individuals, from across the country, participated in this discussion and we would like to acknowledge their considerable efforts. The syntax of the standard was originally specified in the Backus-Naur Form (BNF) meta-language. Ken L. Harrenstien, of SRI International, was responsible for re-coding the BNF into an augmented BNF which compacts the specification and allows increased comprehensibility.

Preceding page blank



CONTENTS

PREFACE.....	iii
Section	
I. INTRODUCTION.....	1
II. FRAMEWORK.....	2
III. SYNTAX.....	4
A. Notational Conventions.....	4
B. Lexical Analysis of Messages.....	5
C. General Syntax of Messages.....	13
D. Syntax of General Addressee Items.....	15
E. Supporting Constructs.....	15
IV. SEMANTICS.....	17
A. Address Fields.....	17
B. Reference Specification Fields.....	22
C. Other Fields and Syntactic Items.....	23
D. Dates and Times.....	24
V. EXAMPLES.....	25
A. Addresses.....	25
B. Address Lists.....	26
C. Originator Items.....	26
D. Complete Headers.....	28
Appendix	
A. ALPHABETICAL LISTING OF SYNTAX RULES.....	31
B. SIMPLE PARSING.....	35
BIBLIOGRAPHY.....	37

*Preceding page blank*



## I. INTRODUCTION

This standard specifies a syntax for text messages which are passed between computer users within the framework of "electronic mail". The standard supersedes the informal standards specified in ARPANET Request for Comments numbers 561, "Standardizing Network Mail Headers", and 680, "Message Transmission Protocol". In this document, a general framework is first described; the formal syntax is then specified, followed by a discussion of the semantics. Finally, a number of examples are given.

This specification is intended strictly as a definition of what is to be passed between hosts on the ARPANET. It is NOT intended to dictate either features which systems on the Network are expected to support, or user interfaces to message creating or reading programs.

A distinction should be made between what the specification REQUIRES and what it ALLOWS. Messages can be made complex and rich with formally-structured components of information or can be kept small and simple, with a minimum of such information. Also, the standard simplifies the interpretation of differing visual formats in messages. These simplifications facilitate the formal specification and indicate what the OFFICIAL semantics are for messages. Only the visual aspect of a message is affected and not the interpretation of information within it. Implementors may choose to retain such visual distinctions.

---

Preceding page blank

## II. FRAMEWORK

Since there are many message systems which exist outside the ARPANET environment, as well as those within it, it may be useful to consider the general framework, and resulting capabilities and limitations, provided by this standard.

Messages are expected to consist of lines of text. No special provisions are made, at this time, for encoding drawings, facsimile, speech, or structured text.

No significant consideration has been given to questions of data compression or transmission/storage efficiency. The standard, in fact, tends to be very free with the number of bits consumed. For example, field names are specified as free text, rather than special terse codes.

A general "memo" framework is used. That is, a message consists of some information, in a rigid format, followed by the main part of the message, which is text and whose format is not specified in this document. The syntax of several fields of the rigidly-formatted ("header") section is defined in this specification; some of the header fields must be included in all messages. The syntax which distinguishes between headers is specified separately from the internal syntax for particular headers. This separation is intended to allow extremely simple parsers to operate on the overall structure of messages, without concern for the detailed structure of individual headers. Appendix B is provided to facilitate construction of these simple parsers. In addition to the fields specified in this document, it is expected that other fields will gain common use. User-defined header fields allow systems to extend their functionality while maintaining a uniform framework. The approach is similar to that of the TELNET protocol, in that a basic standard is defined which includes a mechanism for (optionally) extending itself. As necessary, the authors of this document will regulate the publishing of specifications for these "extension-fields", through the same mechanisms used to publish this document.

Such a framework severely constrains document tone and appearance and is primarily useful for most intra-organization communications and relatively structured inter-organization communication. A more robust environment might allow for multi-font, multi-color, multi-dimension encoding of information. A less robust environment, as is present in most single-machine message systems, would more severely constrain the ability to add fields and the decision to include specific fields. In contrast to paper-based communication, it is interesting to note that the

RECEIVER of a message can exercise an extraordinary amount of control over the message's appearance. The amount of actual control available to message receivers is contingent upon the capabilities of their individual message systems.

### III. SYNTAX

This syntax is given in five parts. The first part describes the notation used in the specification. The second part describes the base-level lexical analyzers which feed the higher-level parser described in the succeeding sections. The third part gives a general syntax for messages and standard header fields; and the fourth part specifies the syntax of addresses. A final part specifies some general syntax which supports the other sections.

#### A. NOTATIONAL CONVENTIONS

These specifications are made in an augmented Backus-Naur Form (BNF). Differences from standard BNF involve the naming of rules, the indication of repetition and of "local" alternatives.

##### 1. Rule naming

Angle brackets ("<", ">") are not used, in general. The name of a rule is simply the name itself, rather than "<name>". Quotation-marks enclose literal text (which may be upper and/or lower case). Certain basic rules are in uppercase, such as SPACE, TAB, CRLF, DIGIT, ALPHA, etc. Angle brackets are used in rule definitions, and in the rest of this document, whenever their presence will facilitate discerning the use of rule names.

##### 2. Parentheses: Local alternatives

Elements enclosed in parentheses are treated as a single element. Thus, "(elem (foo / bar) elem)" allows "(elem foo elem)" and "(elem bar elem)".

##### 3. \* construct: Repetition

The character "\*" preceding an element indicates repetition. The full form is:

<l>\*<m>element

indicating at least <l> and at most <m> occurrences of element. Default values are 0 and infinity so that "\* (element)" allows any number, including zero; "1\*element" requires at least one; and "1\*2element" allows one or two.

## III. Syntax

## A. Notational Conventions

## 4. &lt;number&gt;element :

"<n>(element)" is equivalent to "<n>\*<n>(element)"; that is, exactly <n> occurrences of (element). Thus 2DIGIT is a 2-digit number, and 3ALPHA is a string of three alphabetic characters.

## 5. # construct: Lists

A construct "#" is defined, similar to "\*", as follows:

<1>#<m>element

indicating at least <1> and at most <m> elements, each separated by one or more commas (","). This makes the usual form of lists very easy; a rule such as '(element \*(", " element))' can be shown as "1#element". Wherever this construct is used, null elements are allowed, but do not contribute to the count of elements present. That is, "(element),,(element)" is permitted, but counts as only two elements. Therefore, where at least one element is required, at least one non-null element must be present.

## 6. [optional]

Square brackets enclose optional elements; "[foo bar]" is equivalent to "\*1(foo bar)".

## 7. ; Comments

A semi-colon, set off some distance to the right of rule text, starts a comment which continues to the end of line. This is a simple way of including useful notes in parallel with the specifications.

## B. LEXICAL ANALYSIS OF MESSAGES

## 1. General Description

A message consists of headers and, optionally, a body (i.e. a series of text lines). The text part is just a sequence of lines containing ASCII characters; it is separated from the headers by a null line (i.e., a line with nothing preceding the CRLF).

## III. Syntax

## B. Lexical Analysis

## a. Folding and unfolding of headers

Each header item can be viewed as a single, logical line of ASCII characters. For convenience, the field-body portion of this conceptual entity can be split into a multiple-line representation (i.e., "folded"). The general rule is that wherever there can be linear-white-space (NOT simply LWSP-chars), a CRLF immediately followed by AT LEAST one LWSP-char can instead be inserted. (However, a header's name and the following colon (":"), which occur at the beginning of the header item, may NOT be folded onto multiple lines.) Thus, the single line

To: "Joe Dokes & J. Harvey" <ddd at Host>, JJV at BBN

can be represented as

To: "Joe Dokes & J. Harvey" <ddd at Host>,  
JJV at BBN

and

To: "Joe Dokes & J. Harvey"  
<ddd at Host>,  
JJV at BBN

and

To: "Joe Dokes  
& J. Harvey" <ddd at Host>, JJV at BBN

The process of moving from this folded multiple-line representation of a header field to its single line representation will be called "unfolding". Unfolding is accomplished by regarding CRLF immediately followed by a LWSP-char as equivalent to the LWSP-char.

## b. Structure of header fields

Once header fields have been unfolded, they may be viewed as being composed of a field-name followed by a colon (":"), followed by a field-body. The field-name must be composed of printable ASCII characters (i.e., characters which have values between 33. and 126., decimal, except colon) and LWSP-chars. The field-body may be composed of any ASCII characters (other than an unquoted CRLF, which has been removed by unfolding).

Certain field-bodies of header fields may be interpreted according to an internal syntax which some systems may wish to parse. These fields will be referred to as "structured" fields. Examples include fields containing dates and

## III. Syntax

## B. Lexical Analysis

addresses. Other fields, such as "Subject" and "Comments", are regarded simply as strings of text.

NOTE: Field-names, unstructured field bodies and structured field bodies each are scanned by their own, INDEPENDENT "lexical" analyzer.

## c. Field-names

To aid in the creation and reading of field-names, the free insertion of LWSP-chars is allowed in reasonable places.

Rather than obscuring the syntax specification for field-name with the explicit syntax for these LWSP-chars, the existence of a "lexical" analyzer is assumed. The analyzer interprets the text which comprises the field-name as a sequence of field-name atoms (fnatoms) separated by LWSP-chars

Note that ONLY LWSP-chars may occur between the fnatoms of a field-name and that CRLFs may NOT. In addition, comments are NOT lexically recognized, as such, but parenthesized strings are legal as part of field-names. These constraints are different from what is permissible within structured field bodies. In particular, this means that header field-names must wholly occur on the FIRST line of a folded header item and may NOT be split across two or more lines.

## d. Unstructured field bodies

For some fields, such as "Subject" and "Comments", no structuring is assumed; and they are treated simply as texts, like those in the message body. Rules of folding apply to these fields, so that such field bodies which occupy several lines must therefore have the second and successive lines indented by at least one LWSP-char.

## e. Structured field bodies

To aid in the creation and reading of structured fields, the free insertion of linear-white-space (which permits folding by inclusion of CRLFs) is allowed in reasonable places. Rather than obscuring the syntax specifications for these structured fields with explicit syntax for this linear-white-space, the existence of another "lexical" analyzer is assumed. This analyzer does not apply for field bodies which are simply unstructured strings of text, as described above. It provides an interpretation of the unfolded text comprising the body of the field as a sequence of lexical symbols. These symbols are:

- individual special characters
- quoted-strings

- comments
- atoms

The first three of these symbols are self-delimiting. Atoms are not; they therefore are delimited by the self-delimiting symbols and by linear-white-space. For the purposes of regenerating sequences of atoms and quoted-strings, exactly one SPACE is assumed to exist and should be used between them. (Also, in Section III.B.3.a, note the rules concerning treatment of multiple contiguous LWSP-chars.)

So, for example, the folded body of an address field

```
:sysmail"@ Some-Host,  
Muhammed(I am the greatest)Ali at(the)WBA
```

is analyzed into the following lexical symbols and types:

" : sysmail "	quoted string
@	special
Some-Host	atom
,	special
Muhammed	atom
(I am the greatest)	comment
Ali	atom
at	atom
(the)	comment
WBA	atom

The canonical representations for the data in these addresses are the following strings (note that there is exactly one SPACE between words):

:sysmail at Some-Host

and

Muhammed Ali at WBA

## 2. Formal Definitions

The first four rules, below, indicate a meta-syntactic for fields, without regard to their particular type or internal syntax. The remaining rules define basic syntactic structures which are used by the rules in Sections III.C, III.D, and III.E.

```
field      = field-name ":" [ field-body ] CRLF  
field-name = fnatom *( LWSP-char [fnatom] )
```

## III. Syntax

## B. Lexical Analysis

```

fnatom      = 1*

```

```
comment      = "(" * (ctext / comment / quoted-pair) ")"
ctext       = <any CHAR excluding "(",      ; => may be folded
              ")" and CR, and including
              linear-white-space>
quoted-pair  = "\" CHAR
```

### 3. Clarifications

#### a. "White space"

Remember that in field-names and structured field bodies, MULTIPLE LINEAR WHITE SPACE TELNET ASCII CHARACTERS (namely HTABs and SPACES) ARE TREATED AS SINGLE SPACES AND MAY FREELY SURROUND ANY SYMBOL. In all header fields, the only place in which at least one space is REQUIRED is at the beginning of continuation lines in a folded field. When passing text to processes which do not interpret text according to this standard (e.g., ARPANET FTP mail servers), then exactly one SPACE should be used in place of arbitrary linear-white-space and comment sequences.

WHEREVER A MEMBER OF THE LIST OF <DELIMITER>S IS ALLOWED, LWSP-CHARS MAY ALSO OCCUR BEFORE AND/OR AFTER IT.

Writers of mail-sending (i.e. header generating) programs should realize that there is no Network-wide definition of the effect of horizontal-tab TELNET ASCII characters on the appearance of text at another Network host; therefore, the use of tabs in message headers, though permitted, is discouraged.

Note that during transmissions across the ARPANET using TELNET NVT connections, data must conform to TELNET NVT conventions (e.g., CR must be followed by either LF, making a CRLF, or <null>, if the CR is to stand alone).

#### b. Comments

Comments are detected as such only within field-bodies of structured fields. A comment is a set of TELNET ASCII characters, which is not within a quoted-string and which is enclosed in matching parentheses; parentheses nest, so that if an unquoted left parenthesis occurs in a comment string, there must also be a matching right parenthesis. When a comment is used to act as the delimiter between a sequence of two lexical symbols, such as two atoms, it is lexically equivalent with one SPACE, for the purposes of regenerating the sequence, such as when passing the sequence onto an FTP mail server.

In particular comments are NOT passed to the FTP server, as part of a MAIL or MLFL command, since comments are not part of the "formal" address.

If a comment is to be "folded" onto multiple lines, then the syntax for folding must be adhered to. (See items III.B.1.a, above, and III.B.3.f, below.) Note that the official semantics therefore do not "see" any unquoted CRLFs which are in comments, although particular parsing programs may wish to note their presence. For these programs, it would be reasonable to interpret a "CRLF LWSP-char" as being a CRLF which is part of the comment; i.e., the CRLF is kept and the LWSP-char is discarded. Quoted CRLFs (i.e., a backslash followed by a CR followed by a LF) still must be followed by at least one LWSP-char.

c. Delimiting and quoting characters

The quote character (backslash) and characters which delimit syntactic units are not, generally, to be taken as data which are part of the delimited or quoted unit(s). The one exception is SPACE. In particular, the quotation-marks which define a quoted-string, the parentheses which define a comment and the backslash which quotes a following character are NOT part of the quoted-string, comment or quoted character. A quotation-mark which is to be part of a quoted-string, a parenthesis which is to be part of a comment and a backslash which is to be part of either must each be preceded by the quote-character backslash ("\""). Note that the syntax allows any character to be quoted within a quoted-string or comment; however only certain characters MUST be quoted to be included as data. These characters are those which are not part of the alternate text group (i.e., ctext or qtext).

A single SPACE is assumed to exist between contiguous words in a phrase, and this interpretation is independent of the actual number of LWSP-chars which the creator places between the words. To include more than one SPACE, the creator must make the LWSP-chars be part of a quoted-string.

Quotation marks which delimit a quoted string and backslashes which quote the following character should NOT accompany the quoted-string when the string is used with processes that do not interpret data according to this specification (e.g., ARPANET FTP mail servers).

d. Quoted-strings

Where permitted (i.e., in words in structured fields) quoted-strings are treated as a single symbol (i.e. equivalent to an atom, syntactically). If a quoted-string is to be "folded" onto multiple lines, then the syntax for folding must be adhered to. (See items III.B.1.a, above, and III.B.3.f, below.) Note that the official semantics therefore do not "see" any bare CRLFs which are in quoted-strings, although particular parsing programs may wish to note their presence. For these programs, it would be reasonable to interpret a "CRLF LWSP-char" as being a CRLF which is part of the quoted-string; i.e., the CRLF is kept and the LWSP-char is discarded. Quoted CRLFs (i.e., a backslash followed by a CR followed by a LF) are also subject to rules of folding, but the presence of the quoting character (backslash) explicitly indicates that the CRLF is data to the quoted string. Stripping off the first following LWSP-char is also appropriate when parsing quoted CRLFs.

e. Bracketing characters

There are three types of brackets which must be well nested:

- o Parentheses are used to indicate comments.
- o Angle brackets ("<" and ">") are generally used to indicate the presence of at least one machine-readable code (e.g., delimiting mailboxes).
- o Colon/semi-colon (":" and ";") are used in address specifications to indicate that the included list of addresses are to be treated as a group.

f. Case independence of certain specials atoms

Certain atoms, which are represented in the syntax as literal alphabetic strings, can be represented in any combination of upper and lower case. These are:

- field-name,
- "Include", "Postal" and equivalent atoms in a ":"<atom>": " address specification,
- "at", in a host-indicator,
- node,
- day-of-week,
- month, and
- zones.

When matching an atom against one of these literals, case is to be ignored. For example, the field-names "From", "FROM",

## III. Syntax

## B. Lexical Analysis

"from", and even "From" should all be treated identically. However, the case shown in this specification is suggested for message-creating processes. Note that, at the level of this specification, case IS relevant to other words and texts. Also see Section IV.A.1.f, below.

## g. Folding long lines

Each header item (field of the message) may be represented on exactly one line consisting of the name of the field and its body; this is what the parser sees. For readability, it is recommended that the field-body portion of long header items be "folded" onto multiple lines of the actual header. "Long" is commonly interpreted to mean greater than 65 or 72 characters. The former length is recommended as a limit, but it is not imposed by this standard.

## h. Backspace characters

Backspace TELNET ASCII characters (ASCII BS, decimal 8.) may be included in texts and quoted-strings to effect overstriking; however, any use of backspaces which effects an overstrike to the left of the beginning of the text or quoted-string is prohibited.

## C. GENERAL SYNTAX OF MESSAGES:

NOTE: Due to an artifact of the notational conventions, the syntax indicates that, when present, "Date", "From", "Sender", and "Reply-To" fields must be in a particular order. These header items must be unique (occur exactly once). However header fields, in fact, are NOT required to occur in any particular order, except that the message body must occur AFTER the headers. For readability and ease of parsing by simple systems, it is recommended that headers be sent in the order "Date", "From", "Subject", "Sender", "To", "cc", etc. This specification permits multiple occurrences of most optional-fields. However, their interpretation is not specified here, and their use is strongly discouraged.

The following syntax for the bodies of various fields should be thought of as describing each field body as a single long string (or line). The section on Lexical Analysis (section II.B) indicates how such long strings can be represented on more than one line in the actual transmitted message.

```
message      = fields *( CRLF *text )          ; Everything after
                           ; first null line
                           ; is message body

fields       = date-field                      ; Creation time-stamp
                           originator-fields
                           *optional-field
                           ; & author id are
                           ; required: others
                           ; are all optional

originator-fields =
    ( "From"      ":" mailbox      ; Single author
      ["Reply-To" ":" #address] )
  / ( "From"      ":" 1#address   ; Multiple authors &
      "Sender"     ":" mailbox    ; may have non-mach-
      ["Reply-To" ":" #address] ) ; ine addresses

date-field   = "Date"      ":" date-time

optional-field =
    "To"         ":" #address
  / "cc"        ":" #address
  / "bcc"       ":" #address      ; Blind carbon
  / "Subject"   ":" *text
  / "Comments"  ":" *text
  / "Message-ID" ":" mach-id    ; Only one allowed
  / "In-Reply-To" ":" #(phrase / mach-id)
  / "References" ":" #(phrase / mach-id)
  / "Keywords"  ":" #phrase
  / extension-field           ; To be defined in
                           ; supplemental
                           ; specifications
  / user-defined-field        ; Must have unique
                           ; field-name & may
                           ; be pre-empted

extension-field = <Any field which is defined in a document
                  published as a formal extension to this
                  specification>

user-defined-field = <Any field which has not been defined in
                     this specification or published as an extension to
                     this specification; names for such fields must be
                     unique and may be preempted by published
                     extensions>
```

## III. Syntax

## D. Addressee Items

## D. SYNTAX OF GENERAL ADDRESSEE ITEMS

```

address      = host-phrase                                ; Machine mailbox
              / ([phrase] "<" #address ">")          ; Individual / List
              / ([phrase] ":" #address ";")           ; Group
              / quoted-string                          ; Arbitrary text
              / (":" ( "Include"
                      / "Postal"
                      / atom)
                  ":" address)                     ; Extended data type

mailbox      = host-phrase / (phrase mach-id)

mach-id      = "<" host-phrase ">"                   ; Contents must never
              ; be modified!

```

## E. SUPPORTING CONSTRUCTS

```

host-phrase = phrase host-indicator ; Basic address

host-indicator = 1*( ("at" / "@") node ) ; Right-most node is
                                              ; at top of network
                                              ; hierarchy; left-
                                              ; most must be host

node        = word / 1*DIGIT          ; Official host or
                                              ; network name or
                                              ; decimal address

date-time   = [ day-of-week "," ] date time

day-of-week = "Monday"    / "Mon"     / "Tuesday"   / "Tue"
              / "Wednesday" / "Wed"     / "Thursday" / "Thu"
              / "Friday"     / "Fri"     / "Saturday" / "Sat"
              / "Sunday"     / "Sun"

date        = 1*2DIGIT ["-"] month      ; day month year
              ["-"] (2DIGIT / 4DIGIT)       ; e.g. 20 Aug [19] 77

month       = "January"   / "Jan"     / "February" / "Feb"
              / "March"     / "Mar"     / "April"     / "Apr"
              / "May"       / "June"    / "July"      / "Jun"
              / "July"      / "Jul"     / "August"   / "Aug"
              / "September" / "Sep"     / "October"  / "Oct"
              / "November"  / "Nov"     / "December" / "Dec"

```

## III. Syntax

## E. Supporting Constructs

time	= hour zone	; ANSI and Military ; (seconds optional)
hour	= 2DIGIT [":"] 2DIGIT [ [":"] 2DIGIT ] ; 0000[00] - 2359[59]	
zone	= ( ["-"] ( "GMT" / "NST" / / "AST" / "ADT" / "EST" / "EDT" / "CST" / "CDT" / "MST" / "MDT" / "PST" / "PDT" / "YST" / "YDT" / "HST" / "HDT" / "BST" / "BDT" 1ALPHA )) / ( ("+" / "-") 4DIGIT )	; Relative to GMT: ; North American ; Newfoundland: -3:30 ; Atlantic: -4/-3 ; Eastern: -5/-4 ; Central: -6/-5 ; Mountain: -7/-6 ; Pacific: -8/-7 ; Yukon: -9/-8 ; Haw/Ala -10/-9 ; Bering: -11/-10 ; Military: Z = GMT; ; A:-1; (J not used) ; M:-12; N:+1; Y:+12 ; Local differential ; hours/min. (HHMM)
phrase	= 1*word	; Sequence of words. ; Separation seman- ; tically = SPACE
word	= atom / quoted-string	

## IV. Semantics

## A. Address Fields

## IV. SEMANTICS

## A. ADDRESS FIELDS

## 1. General

- a. The phrase part of a host-phrase in an address specification (i.e., the host's name for the mailbox) is understood to be whatever the receiving FTP Server allows (for example, TENEX systems do not now understand addresses of the form "P. D. Q. Bach", but another system might).

Note that a mailbox is a conceptual entity which does not necessarily pertain to file storage. For example, some sites may choose to print mail on their line printer and deliver the output to the addressee's desk.

An individual may have several mailboxes and a group of individuals may wish to receive mail as a single unit (i.e., a distribution list). The second and third alternatives of an address list (#address) allow naming a collection of subordinate addresses list(s). Recipient mailboxes are specified within the bracketed part ("<" - ">" or ":" - ";") of such named lists. The use of angle-brackets ("<", ">") is intended for the cases of individuals with multiple mailboxes and of special mailbox lists; it is not expected to be nested more than one level, although the specification allows such nesting. The use of colon/semi-colon (":", ";") is intended for the case of groups. Groups can be expected to nest (i.e., to contain subgroups). For both individuals and groups, a copy of the transmitted message is to be sent to EACH mailbox listed. For the case of a special list, treatment of addresses is defined in the relevant subsections of this section.

- b. The inclusion of bare quoted-strings as addresses (i.e., the fourth address-form alternative) is allowed as a syntactic convenience, but no semantics are defined for their use. However, it is reasonable, when replicating an address list, to replicate ALL of its members, including quoted-strings.
- c. ":Include:" specifications are used to refer to one or more locations containing stored address lists (#address). If more than one location is referenced, the address part of the Include phrase must be a list (#address) surrounded by angle-brackets, as per the "Individual / List" alternative of <address>. Constituent addresses must resolve to a host-

IV. Semantics  
A. Address Fields

phrase; only they have any meaning within this construct. The phrase part of indicated host-phrases should contain text which the referenced host can resolve to a file. This standard is not a protocol and so does not prescribe HOW data is to be retrieved from the file. However, the following requirements are made:

- o The file must be accessible through the local operating system interface (if it exists), given adequate user access rights; and
- o If a host has an FTP server and a user is able to retrieve any files from the host using that server, then the file must be accessible through FTP, using DEFAULT transfer settings, given adequate user access rights.

It is intended that this mechanism allow programs to retrieve such lists automatically.

The interpretation of such a file reference follows. This is not intended to imply any particular implementation scheme, but is presented to aid in understanding the notion of including file contents in address lists:

- o Elements of the address list part are alternates and the contents of ONLY ONE of them are to be included in the resultant address list.
- o The contents of the file indicated by a member host-phrase are treated as an address list and are inserted as an address list (#address) in the position of the path item in the syntax. That is, the TELNET ASCII characters specifying the entire `<address>` is replaced by the contents of one of the files to which the host-phrase(s), of the address list (#address), refers. Therefore, the contents of each file, indicated by an Include address, must be syntactically self-contained and must adhere to the full syntax prescribed herein for an address list.
- d. `":Postal:"` specifications are used to indicate (U.S.) postal addresses, but can be treated the same as quoted-string addresses. To reference a list of postal addresses, the list must conform to the "Individual / List" alternative of `<address>`. The `":Include:"` alternative also is valid.
- e. The `"':'"` atom `'::'` syntax is intended as a general mechanism for indicating specially data-typed addresses. As with extension-fields, the authors of this document will regulate

## IV. Semantics

## A. Address Fields

the publishing of specifications for these extended data-types. In the absence of defined semantics, any occurrence of an address in this form may be treated as a quoted-string address.

- f. A node name must be THE official name of a network or a host, or else a decimal number indicating the Network address for that network or host, at the time the message is created. The USE OF NUMBERS IS STRONGLY DISCOURAGED and is permitted only due to the occasional necessity of bypassing local name tables. For the ARPANET, official names are maintained by the Network Information Center at SRI International, Menlo Park, California.

Whenever a message might be transmitted or migrate to a host on another network, full hierarchical addresses must be specified. These are indicated as a series of words, separated by at-sign or "at" indications. The communication environment is assumed to consist of a collection of networks organized as independent "trees" except for connections between the root nodes. That is, only the roots can act as gateways between these independent networks. While other actual connections may exist, it is believed that presuming this type of organization will provide a reliable method for describing VALID, if not EFFICIENT, paths between hosts. A typical full mailbox specification might therefore look like:

Friendly User @ hosta @ local-net1 @ major-netq

In the simplest case, a mail-sending host should transmit the message to the node which is mentioned last (farthest to the right), strip off that node reference from the specification, and then pass the remaining host-phrase to the recipient host (in the ARPANET, its FTP server) for it to process. This treats the remaining portion of the host-indicator merely as the terminating part of the phrase.

NOTE: When passing any portion of a host-indicator onto a process which does not interpret data according to this standard (e.g., ARPANET FTP servers), "@" must be used and not "at" and it must not be preceded or followed by any LWSP-chars. Using the above example, the following string would be passed to the major-netq gateway:

Friendly User@hosta@local-net1

When the sending host has more knowledge of the network environment, then it should send the message along a more efficient path, making appropriate changes to the form of the host-phrase which it gives to the recipient host.

IV. Semantics  
A. Address Fields

To use the above specification as an example: If a sending hostb also were part of local-net1, then it could send the message directly to hosta and would give only the phrase "Friendly User" to hosta's mail-receiving program. If hostb were part of local-net2, along with hostc, and happened to know that hosta and hostc were part of another local-net, then hostb could send the message to hostc to the address "Friendly User@hosta".

The phrase in a host-phrase is intended to be meaningful only to the indicated receiving host. To all other hosts, the phrase is to be treated as an uninterpreted string. No case transformations should be (automatically) performed on the phrase. The phrase is passed to the local host's mail sending program; it is the responsibility of the destination host's mail receiving (distribution) program to perform case mapping on this phrase, if required, to deliver the mail.

2. Originator Fields

**WARNING:** The standard allows only a subset of the combinations possible with the From, Sender, and Reply-To fields. The limitation is intentional.

a. From

This field contains the identity of the person(s) who wished this message to be sent. The message-creation process should default this field to be a single machine address, indicating the AGENT (person or process) entering the message. If this is NOT done, the "Sender" field MUST be present; if this IS done, the "Sender" field is optional.

b. Sender

This field contains the identity of the AGENT (person or process) who sends the message. It is intended for use when the sender is not the author of the message, or to indicate who among a group of authors actually sent the message. If the contents of the "Sender" field would be completely redundant with the "From" field, then the "Sender" field need not be present and its use is discouraged (though still legal); in particular, the "Sender" field MUST be present if it is NOT the same as the "From" Field.

The Sender host-phrase includes a phrase which must correspond to a specific agent (i.e., a human user or a computer program) rather than a standard address. This indicates the expectation that the field will identify the single AGENT (person or process) responsible for sending the

## IV. Semantics

## A. Address Fields

mail and not simply include the name of a mailbox from which the mail was sent. For example in the case of a shared login name, the name, by itself, would not be adequate. The phrase part of the host-phrase, which refers to this agent, is expected to be a computer system term, and not (for example) a generalized person reference which can be used outside the network text message context.

Since the critical function served by the "Sender" field is the identification of the agent responsible for sending mail and since computer programs cannot be held accountable for their behavior, is strongly recommended that when a computer program generates a message, the HUMAN who is responsible for that program be referenced as part of the "Sender" field host-phrase.

## c. Reply-To

This field provides a general mechanism for indicating any mailbox(es) to which responses are to be sent. Three typical uses for this feature can be distinguished. In the first case, the author(s) may not have regular machine-based mailboxes and therefore wish(es) to indicate an alternate machine address. In the second case, an author may wish additional persons to be made aware of, or responsible for, responses; responders should send their replies to the "Reply-To" mailbox(es) listed in the original message. A somewhat different use may be of some help to "text message teleconferencing" groups equipped with automatic distribution services: include the address of that service in the "Reply-To" field of all messages submitted to the teleconference; then participants can "reply" to conference submissions to guarantee the correct distribution of any submission of their own.

Reply-To fields are NOT required to contain any machine addresses (i.e., host-phrases). Note, however, that the absence of even one valid network address will tend to prevent software systems from automatically assisting users in conveniently responding to mail.

NOTE: For systems which automatically generate address lists for replies to messages, the following recommendations are made:

- o The receiver, when replying to a message, should NEVER automatically include the "Sender" host-phrase in the reply's address list;
- o If the "Reply-To" field exists, then the reply should go ONLY to the addresses indicated in that field and not to the addresses indicated in the "From" field.

(Extensive examples are provided in Section V.) This recommendation is intended only for originator-fields and is not intended to suggest that replies should not also be sent to the other recipients of this message. It is up to the respective mail handling programs to decide what additional facilities will be provided.

3. Receiver Fields

a. To

This field contains the identity of the primary recipients of the message.

b. cc

This field contains the identity of the secondary recipients of the message.

b. Bcc

This field contains the identity of additional recipients of the message. The contents of this field are not included in copies of the message sent to the primary and secondary recipients. Some systems may choose to include the text of the "Bcc" field only in the author(s)'s copy, while others may also include it in the text sent to all those indicated in the "Bcc" list.

B. REFERENCE SPECIFICATION FIELDS

1. Message-ID

This field contains a unique identifier (the phrase) which refers to THIS version of THIS message. The uniqueness of the message identifier is guaranteed by the host which generates it. This identifier is intended to be machine readable and not necessarily meaningful to humans. A message identifier pertains to exactly one instantiation of a particular message; subsequent revisions to the message should each receive a new message identifier.

2. In-Reply-To

The contents of this field identify previous correspondence which this message answers. Note that if message identifiers are used in this field, they must use the mach-id specification format.

3. References

The contents of this field identify other correspondence which this message references. Note that if message identifiers are used, they must use the mach-id specification format.

4. Keywords

This field contains keywords or phrases, separated by commas.

C. OTHER FIELDS AND SYNTACTIC ITEMS

1. Subject

The "Subject" field is intended to provide as much information as necessary to adequately summarize or indicate the nature of the message.

2. Comments

Permits adding text comments onto the message without disturbing the contents of the message's body.

3. Extension-field

A relatively limited number of common fields have been defined in this document. As network mail requirements dictate, additional fields may be standardized. The authors of this document will regulate the publishing of such definitions as extensions to the basic specification.

4. User-defined-field

Individual users of network mail are free to define and use additional header fields. Such fields must have names which are not already used in the current specification or in any definitions of extension-fields, and the overall syntax of these user-defined-fields must conform to this specification's rules for delimiting and folding fields. Due to the extension-field publishing process, the name of a user-defined-field may be preempted.

## IV. Semantics

## D. Dates

## D. DATES AND TIMES

If included, day-of-week must be the day implied by the date specification.

Time zone may be indicated in several ways. The military standard uses a single character for each zone. "Z" is Greenwich Mean Time; "A" indicates one hour earlier, and "M" indicates 12 hours earlier; "N" is one hour later, and "Y" is 12 hours later. The letter "J" is not used. The other remaining two forms are taken from ANSI standard X3.51-1975. One allows explicit indication of the amount of offset from GMT; the other uses common 3-character strings for indicating time zones in North America.

## V. EXAMPLES

### A. ADDRESSES

1. Alfred E. Neuman <Neuman at BBN-TENEXA>
2. Neuman@BBN-TENEXA

These two "Alfred E. Neuman" examples have identical semantics, as far as the operation of the local host's mail sending (distribution) program (also sometimes called its "mailer") and the remote host's FTP server are concerned. In the first example, the "Alfred E. Neuman" is ignored by the mailer, as "Neuman at BBN-TENEXA" completely specifies the recipient. The second example contains no superfluous information, and, again, "Neuman@BBN-TENEXA" is the intended recipient.

#### 3. Al Neuman at BBN-TENEXA

This is identical to "Al Neuman <Al Neuman at BBN-TENEXA>". That is, the full phrase, "Al Neuman", is passed to the FTP server. Note that not all FTP servers accept multi-word identifiers; and some that do accept them will treat each word as a different addressee (in this case, attempting to send a copy of the message to "Al" and a copy to "Neuman").

#### 4. "George Lovell, Ted Hackle" <Shared-Mailbox at Office-1>

This form might be used to indicate that a single mailbox is shared by several users. The quoted string is ignored by the originating host's mailer, as "Shared-Mailbox at Office-1" completely specifies the destination mailbox.

#### 4. Wilt (the Stilt) Chamberlain at NBA

The "(the Stilt)" is a comment, which is NOT included in the destination mailbox address handed to the originating system's mailer. The address is the string "Wilt Chamberlain", with exactly one space between the first and second words. (The quotation marks are not included.)

B. ADDRESS LISTS

Gourmets: Pompous Person <WhoZiWhatZit at Cordon-Bleu>;  
Cooks: Childs at WGBH, Galloping Gourmet at  
ANT (Australian National Television);,  
Wine Lovers: Cheapie at Discount-Liquors,  
Port at Portugal;;,  
Jones at SEA

This group list example points out the use of comments, the nesting of groups, and the mixing of addresses and groups. Note that the two consecutive semi-colons preceding "Jones at SEA" mean that Jones is NOT a member of the Gourmets group.

C. ORIGINATOR ITEMS

1. Author-sent

George Jones logs into his Host as "Jones". He sends mail himself.

From: Jones at Host  
or  
From: George Jones <Jones at Host>

2. Secretary-sent

George Jones logs in as Jones on his Host. His secretary, who logs in as Secy on Shost sends mail for him. Replies to the mail should go to George, of course.

From: George Jones <Jones at Host>  
Sender: Secy at SHost

3. Shared directory or unrepresentative directory-name

George Jones logs in as Group at Host. He sends mail himself; replies should go to the Group mailbox.

From: George Jones <Group at Host>

4. Secretary-sent, for user of shared directory

George Jones' secretary sends mail for George in his capacity as a member of Group while logged in as Secy at Host. Replies should go to Group.

From: George Jones<Group at Host>  
Sender: Secy at Host

Note that there need not be a space between "Jones" and the "<", but adding a space enhances readability (as is the case in other examples).

5. Secretary acting as full agent of author

George Jones asks his secretary (Secy at Host) to send a message for him in his capacity as Group. He wants his secretary to handle all replies.

From: George Jones <Group at Host>  
Sender: Secy at Host  
Reply-To: Secy at Host

6. Agent for user without online mailbox

A non-ARPANET user friend of George's, Sarah, is visiting. George's secretary sends some mail to a friend of Sarah in computer-land. Replies should go to George, whose mailbox is Jones at Host.

From: Sarah Friendly  
Sender: Secy at Host  
Reply-To: Jones at Host

7. Sent by member of a committee

George is a member of a committee. He wishes to have any replies to his message go to all committee members.

From: George Jones  
Sender: Jones at Host  
Reply-To: Big-committee: Jones at Host,  
                                 Smith at Other-Host,  
                                 Doe at Somewhere-Else;

Note that if George had not included himself in the enumeration of Big-committee, he would not have gotten an implicit reply; the presence of the "Reply-to" field SUPERSEDES the sending of a reply to the person named in the "From" field.

8. Example of INCORRECT use

George desires a reply to go to his secretary; therefore his secretary leaves his mailbox address off the "From" field, leaving only his name, which is not, itself, a mailbox address.

From: George Jones  
Sender: Secy at SHost

THIS IS NOT PERMITTED. Replies are NEVER implicitly sent to the "Sender"; George's secretary should have used the "Reply-To" field, or the mail creating program should have forced the secretary to.

9. Agent for member of a committee

George's secretary sends out a message which was authored jointly by all the members of the "Big-committee".

From: Big-committee: Jones at Host,  
Smith at Other-Host,  
Doe at Somewhere-Else;  
Sender: Secy at SHost

D. COMPLETE HEADERS

1. Minimum required:

Date: 26 August 1976 1429-EDT  
From: Jones at Host

2. Using some of the additional fields:

Date: 26 August 1976 1430-EDT  
From: George Jones<Group at Host>  
Sender: Secy at SHOST  
To: Al Neuman at Mad-Host,  
Sam Irving at Other-Host  
Message-ID: <some string at SHOST>

3. About as complex as you're going to get:

Date : 27 Aug 1976 0932-PDT  
From : Ken Davis <KDavis at Other-Host>  
Subject : Re: The Syntax in the RFC  
Sender : KSecy at Other-Host  
Reply-To : Sam Irving at Other-Host  
To : George Jones <Group at Host>, Al Neuman at Mad-Host  
cc : Important folk:  
Tom Softwood <Balsa at Another-Host>,  
Sam Irving at Other-Host;;,  
Standard Distribution::Include:  
  </main/davis/people/standard at Other-Host,  
  "<Jones>standard.dist.3" at Tops-20-Host>,  
  (The following Included Postal list is part  
  of Standard Distribution.)  
  :Postal::Include: Non-net-addrs@Other-host;;  
  :Postal: "Sam Irving, P.O. Box 001, Las Vegas,  
    Nevada" (So that he can stay  
    apprised of the situation)  
Comment : Sam is away on business. He asked me to handle  
his mail for him. He'll be able to provide a  
more accurate explanation when he returns  
next week.  
In-Reply-To: <some string at SHOST>  
Special (action): This is a sample of multi-word field-  
names, using a range of characters. There  
could also be a field-name "Special (info)".  
Message-ID: <4231.629.XYzi-What at Other-Host>



## APPENDIX

### A. ALPHABETICAL LISTING OF SYNTAX RULES

```
address      = host-phrase / quoted-string
              / (*phrase "<" #address ">" )
              / (*phrase ":" #address ";" )
              / (":" ("Include" / "Postal" / atom) ":" address)
ALPHA        = <any TELNET ASCII alphabetic character>
atom         = 1*<any CHAR except specials and CTLs>

CHAR          = <any TELNET ASCII character>
comment       = "(" *(ctext / comment / quoted-pair) ")"
CR           = <TELNET ASCII carriage return>
CRLF          = CR LF
ctext         = <any CHAR excluding "(", ")", CR, LF and
                  including linear-white-space>
CTL           = <any TELNET ASCII control character and DEL>

date          = 1*2DIGIT ["-"] month ["-"] (2DIGIT /4DIGIT)
date-field    = "Date"      ":" date-time
date-time     = [ day-of-week "," ] date time
day-of-week   = "Monday"   / "Mon"   / "Tuesday" / "Tue"
              / "Wednesday" / "Wed"   / "Thursday" / "Thu"
              / "Friday"    / "Fri"   / "Saturday" / "Sat"
              / "Sunday"    / "Sun"
delimiters   = specials / comment / linear-white-space
DIGIT         = <any TELNET ASCII digit>

extension-field = <Any field which is defined in a document
                  published as a formal extension to this
                  specification>

field          = field-name ":" [ field-body ] CRLF
fields         = date-field originator-fields *optional-field
field-body     = field-body-contents
                  [CRLF LWSP-char field-body]
field-body-contents = <the TELNET ASCII characters making up the
                      field-body, as defined in the following sections,
                      and consisting of combinations of atom, quoted-
                      string, and specials tokens, or else consisting of
                      texts>
field-name    = fnatom *(LWSP-char [fnatom])
fnatom        = 1*<any CHAR, excluding CTLs, SPACE, and ":">
```

Preceding page blank

```
host-indicator = 1*( ("at" / "@") node )
host-phrase = phrase host-indicator
hour = 2DIGIT [":"] 2DIGIT [":"] 2DIGIT ]
HTAB = <TELNET ASCII horizontal-tab>
LF = <TELNET ASCII linefeed>
linear-white-space = 1*([CRLF] LWSP-char)
LWSP-char = SPACE / HTAB

mach-id = "<" host-phrase ">"
mailbox = host-phrase / (phrase mach-id)
message = fields *(CRLF *text)
month = "January" / "Jan" / "February" / "Feb"
       / "March" / "Mar" / "April" / "Apr"
       / "May" / "June" / "Jun"
       / "July" / "Jul" / "August" / "Aug"
       / "September" / "Sep" / "October" / "Oct"
       / "November" / "Nov" / "December" / "Dec"

node = word / 1*DIGIT

optional-field =
    "To" ":" #address
   / "cc" ":" #address
   / "bcc" ":" #address
   / "Subject" ":" *text
   / "Comments" ":" *text
   / "Message-ID" ":" mach-id
   / "In-Reply-To" ":" #(phrase / mach-id)
   / "References" ":" #(phrase / mach-id)
   / "Keywords" ":" #phrase
   / extension-field
   / user-defined-field
originator-fields =
    ( "From" ":" mailbox
      ["Reply-To" ":" #address] )
   / ( "From" ":" 1#address
      "Sender" ":" mailbox
      ["Reply-To" ":" #address] )

phrase = 1*word

quoted-pair = "\" CHAR
quoted-string = "<" * (qtext / quoted-pair) ">"
qtext = <any CHAR except "<", CR, or LF and including
      linear-white-space>
SPACE = <TELNET ASCII space>
specials = "(" / ")" / "<" / ">" / "@" / "," / ";" / ":"
           / "\/" / "<">

text = <any CHAR, including bare CR and/or bare LF, but
      NOT including CRLF>
```

time = hour zone

user-defined-field = <Any field which has not been defined in  
this specification or published as an extension to  
this specification; names for such fields must be  
unique and may be preempted by published  
extensions>

word = atom / quoted-string

zone = ( ("+" / "-") 4DIGIT )  
/ ( ["-"] 1ALPHA  
/ "GMT" / "NST" / "AST" / "ADT" / "EST" / "EDT"  
/ "CST" / "CDT" / "MST" / "MDT" / "PST" / "PDT"  
/ "YST" / "YDT" / "HST" / "HDT" / "BST" / "BDT" ))

<"> = <TELNET ASCII quote mark>



## B. SIMPLE PARSING

Some mail-reading software systems may wish to perform only minimal processing, ignoring the internal syntax of structured field-bodies and treating them the same as unstructured-field-bodies. Such software will need only to distinguish:

- Header fields from the message body,
- Beginnings of fields from lines which continue fields,
- Field-names from field-contents.

The abbreviated set of syntactic rules which follows will suffice for this purpose. They describe a limited view of messages and are a subset of the syntactic rules provided in the main part of this specification. One small exception is that the contents of field-bodies consist only of text:

### SYNTAX:

```
message      = *field *(CRLF *text)
field       = field-name ":" [field-body] CRLF
field-name   = fnatom *(LWSP-char [fnatom])
fnatom      = 1*any CHAR, excluding CTLs, SPACE, and ":">
field-body   = *text [CRLF LWSP-char field-body]
```

### SEMANTICS:

Headers occur before the message body and are terminated by a null line (i.e., two contiguous CRLFs).

A line which continues a header field begins with a SPACE or HTAB character, while a line beginning a field starts with a printable character which is not a colon.

A field-name consists of one or more printable characters (excluding colon), each separated by one or more SPACES or HTABS. A field-name MUST be contained on one line. Upper and lower case are not distinguished when comparing field-names.



BIBLIOGRAPHY

- ANSI. Representations of universal time, local time differentials, and United States time zone references for information interchange. ANSI X3.51-1975; American National Standards Institute: New York, 1975.
- Bhushan, A.K. The File Transfer Protocol. ARPANET Request for Comments, No. 354, Network Information Center No. 10596; Augmentation Research Center, Stanford Research Institute: Menlo Park, July 1972.
- Bhushan, A.K. Comments on the File Transfer Protocol. ARPANET Request for Comments, No. 385, Network Information Center No. 11357; Augmentation Research Center, Stanford Research Institute: Menlo Park, August 1972.
- Bhushan, A.K., Pogran, K.T., Tomlinson, R.S., and White, J.E. Standardizing Network Mail Headers. ARPANET Request for Comments, No. 561, Network Information Center No. 18516; Augmentation Research Center, Stanford Research Institute: Menlo Park, September 1973.
- Feinler, E.J. and Postel, J.B. ARPANET Protocol Handbook. Network Information Center No. 7104; Augmentation Research Center, Stanford Research Institute: Menlo Park, April 1976. (NTIS AD A003890).
- McKenzie, A. File Transfer Protocol. ARPANET Request for Comments, No. 454, Network Information Center No. 14333; Augmentation Research Center, Stanford Research Institute: Menlo Park, February 1973.
- McKenzie, A. TELNET Protocol Specification. Network Information Center No. 18639; Augmentation Research Center, Stanford Research Institute: Menlo Park, August 1973.
- Myer, T.H. and Henderson, D.A. Message Transmission Protocol. ARPANET Request for Comments, No. 680, Network Information Center No. 32116; Augmentation Research Center, Stanford Research Institute: Menlo Park, 1975.
- Neigus, N. File Transfer Protocol. ARPANET Request for Comments, No. 542, Network Information Center No. 17759; Augmentation Research Center, Stanford Research Institute: Menlo Park, July 1973.
- Pogran, K., Vittal, J., Crocker, D. and Henderson, A. Proposed official standard for the format of ARPA network messages.

---

Preceding page blank

ARPANET Request for Comments, No. 724, Network Information Center No. 37435; Augmentation Research Center, Stanford Research Institute: Menlo Park, May 1977.

Postel, J.B. Revised FTP Reply Codes. ARPANET Request for Comments, No. 640, Network Information Center No. 30843; Augmentation Research Center, Stanford Research Institute: Menlo Park, June 1974.

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

Robert Bressler, MIT-DMCG  
Richard Guida, MIT-DMCG  
Alex McKenzie, BBN-NET

Obsoletes RFC 360

**REMOTE JOB ENTRY PROTOCOL**



**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

**REMOTE JOB ENTRY PROTOCOL**

**INTRODUCTION**

Remote job entry is the mechanism whereby a user at one location causes a batch-processing job to be run at some other location. This protocol specifies the Network standard procedures for such a user to communicate over the Network with a remote batch-processing server, causing that server to retrieve a job-input file, process the job, and deliver the job's output file(s) to a remote location. The protocol uses a TELNET connection (to a special standardized logger, not socket 1) for all control communication between the user and the server RJE processes. The server-site then uses the File Transfer Protocol to retrieve the job-input file and to deliver the output file(s).

There are two types of users: direct users (persons) and user processes. The direct user communicates from an interactive terminal attached to a TIP or any host. This user may cause the input and/or output to be retrieved/sent on a specific socket at the specified host (such as for card readers or printers on a TIP), or the user may have the files transferred by file-id using File Transfer Protocol. The other type of user is a RJE User-process in one remote host communicating with the RJE Server-process in another host. This type of user ultimately receives its instructions from a human user, but through some unspecified indirect means. The command and response streams of this protocol are designed to be readily used and interpreted by both the human user and the user process.

A particular user site may choose to establish the TELNET control connection for each logical job or may leave the control connection open for extended periods. If the control connection is left open, then multiple job-files may be directed to be retrieved or optionally (to servers that are able to determine the end of one logical job by the input stream and form several jobs out of one input file) one continuous retrieval may be done (as from a TIP card reader). This then forms a "hot" card reader to a particular server with the TELNET connection serving as a "job monitor". Since the output is always transferred job at a time per connection to the output socket, the output from this "hot" reader would appear when ready as if to a "hot" printer. Another possibility for more complex hosts is to attach an RJE User-process to a card reader and take instructions from a lead control card, causing an RJE control TELNET to be opened to the appropriate host with appropriate log-on and input retrieval commands. This card reader would appear to the human user as a Network "hot" card reader. The details of this RJE User-process are beyond the scope of this protocol.

**GENERAL SPECIFICATIONS**

**User**

A human user at a real terminal or a process that supplies the command control stream causing a job to be submitted remotely will be termed the User. The procedure by which a process user receives its instructions is beyond the scope of this protocol.

---

**Preceding page blank**

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

User TELNET

The User communicates its commands over the Network in Network Virtual Terminal code through a User TELNET process in the User's Host. This User TELNET process initiates its activity via ICP to the standard "RJE Logger" socket (socket 5) at the desired RJE-server Host.

RJE-Server TELNET

The RJE-server process receives its command stream from and sends its response stream to the TELNET channel through an RJE-server TELNET process in the server host. This process must listen for the ICP on the "RJE Logger" socket (and cause appropriate ICP socket shifting).

TELNET Connection

The command and response streams for the RJE mechanism are via a TELNET-like connection to a special socket with full specifications according to the current NWG TELNET protocol.

RJE-Server

The RJE-Server process resides in the Host which is providing Remote Batch Job Entry service. This process receives input from the RJE-server TELNET, controls access through the "log-on" procedure, retrieves input job files, queues jobs for execution by the batch system, responds to status inquiries, and transmits job output files when available.

User FTP

All input and output files are transferred under control of the RJE-server process at its initiative. These files may be directly transferred via Request-for-connection to a specific Host/socket or they may be transferred via File Transfer Protocol. If the latter method is used, then the RJE-server acts through its local User FTP process to cause the transfer. This process initiates activity by an active Request-for-connection to the "FTP Logger" in the foreign host.

Server FTP

This process in a remote host (remote from the RJE-server) listens for an ICP from the User FTP and then acts upon the commands from the User FTP causing the appropriate file transfer.

FTP

When File Transfer Protocol is used for RJE files, the standard FTP mechanism is used as fully specified by the current NWG FTProtocol.

RJE Command Language

The RJE system is controlled by a command stream from the User over the TELNET connection specifying the user's identity (log-on), the source of the job input file, the disposition of the job's output files, enquiring about job status, altering job status or output

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

disposition. Additional commands affecting output disposition are includable in the job input file. This command language is explicitly specified in a following section of this protocol.

#### RJE Command Replies

Every command input from the User via TELNET calls for a response message from the RJE-server to the User over the TELNET connection. Certain other conditions also require a response message. These messages are formatted in a standardized manner to facilitate interpretation by both human Users and User processes. A following section of this protocol specifies the response messages.

### RJE COMMANDS OVER TELNET CONNECTION

#### GENERAL CONVENTIONS

1. Each of the commands will be contained in one input line terminated by the standard TELNET "crlf". The line may be of any length desired by the user (explicitly, not restricted to a physical terminal line width). The characters "cr" and "lf" will be ignored by the RJE-server except in the explicit order "crlf" and may be used as needed for local terminal control.
2. All commands will begin with a recognized command name and may then contain recognized syntactic element strings and free-form variable strings (for user-id, file-ids, etc.). Recognized words consist of alphanumeric strings (letters and digits) or punctuation. Recognized alphanumeric string elements must be separated from each other and from unrecognizable strings by at least one blank or a syntactically permitted punctuation. Other blanks may be used freely as desired before or after any syntactic element ("blank" is understood here to mean ASCII SPACE (octal 040); formally:  $\langle \text{blank} \rangle ::= \langle \text{blank} \rangle \langle \text{ASCII SPACE} \rangle \mid \langle \text{ASCII SPACE} \rangle$ ; thus, a sequence of SPACES is also permissible in place of  $\langle \text{blank} \rangle$ , although there is no syntactic necessity for there to be more than one). The "=" after the command name in all commands except OUT and CHANGE is optional.
3. Recognized alphanumeric strings may contain upper case letters or lower case letters in any mixture without syntactic differentiation. Unrecognizable strings will be used exactly as presented with full differentiation of upper and lower case input, unless the host finally using the string defines otherwise.
4. There are two types of Unrecognizable strings: final and imbedded. Final strings appear as the last syntactic element of a command and are parsed as beginning with the next non-blank character of the input stream and continuing to the last non-blank character before the "crlf".

Imbedded strings include "job-id" and "job-file-id" in the OUT, CHANGE, and ALTER commands. At present these fields will be left undelimited since they must only be recognizable by the server host which hopefully can recognize its own job-ids and file-names.

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

**SYNTAX**

The following command descriptions are given in a BNF syntax. Names within angle brackets are non-terminal syntactic elements which are expanded in succeeding syntactic equations. Each equation has the defined name on the left of the ::= and a set of alternative definitions, separated by vertical lines "|", on the right.

**REINITIALIZE**

**REINIT**

This command puts the user into a state identical to the state immediately after a successful connection to the RJE-server, prior to having sent any commands over the TELNET connection. The effective action taken is that of an ABORT and a flushing of all INPUT, OUTPUT and ID information. Naturally, the user is still responsible for any usage charges incurred prior to his REINIT command. The TELNET connection is not affected in any way.

**USER**

User = <user-id>

This command must be the first command over a new TELNET connection. As such, it initiates a "logon" sequence. The response to this command is one of the following:

1. User code in error.
2. Enter password (if user code ok).
3. Log-on ok, proceed (if no password requested).

Another USER command may be sent by the User at any time to change Users. Further input will then be charged to the new user. A server may refuse to honor a new user command if it is not able to process it in its current state (during input file transfer, for example), but the protocol permits the USER command at any time without altering previous activity. An incorrect subsequent USER command or its following PASS command are to be ignored with error response, leaving the original User logged-in.

It is permissible for a server to close the TELNET connection if the initial USER/PASS commands are not completed within a server specified time period. It is not required or implied that the "logged-on" User's user-id be the one used for file transfer or job execution, but only identifies the submitter of the command stream. Servers will establish their own rules relating user-id with the job-execution-user for Job or Output alteration commands.

Successful "log-on" always clears any previous Input or Output default parameters (INID, etc.).

**PASS**

Pass = <password>

This command immediately follows a USER command and completes the "log-on" procedure. Although a particular Server may not require a password and has already

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

indicated "log-on ok" after the USER command, every Server must permit a PASS command (and possibly ignore it) and acknowledge it with a "log-on ok" if the log-on is completed.

BYE

BYE

This command terminates a USER and requests the RJE server to close the TELNET connection. If input transfer is not in progress, the TELNET connection may be closed immediately; if input is in progress, the connection should remain open for result response and then be closed. During the interim, a new USER command (and no other command) is acceptable.

An unexpected close on the TELNET connection will cause the server to take the effective action of an ABORT and a BYE.

INID/INPASS

INID = <user-id>

INPASS = <password>

The specified user-id and password will be sent in the File Transfer request to retrieve the input file. These parameters are not used by the Server in any other way. If this command does not appear, then the USER/PASS parameters are used.

INPATH/INPUT

INPATH = <file-id>

INPUT = <file-id>

INPUT

NOTE: The following syntax will be used for output as well.

```
<file-id> ::= <host-socket> | <host-file>
<host-socket> ::= <host>,<socket><attributes> |
                  <socket><attributes>
                  no <host> part implies the User-site host
<host> ::= <integer>
<socket> ::= <integer>

<integer> ::= D<decimal-integer> | O<octal-integer> |
              H<hexadecimal-integer>
<host-file> ::= <host><attributes>/<pathname>
<attributes> ::= <empty> | :<transmission><code>
<transmission> ::= <empty> | T | A | N

<empty> implies default which is N for Input files
          and A for Output files
T        specifies TELNET-like coding with embedded
          "crlf" for new-line, "ff" for new-page
N        specifies FTP blocked transfer with record
          marks but without other carriage-control
```

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

A specifies FTP blocked records with ASA carriage-control  
(column 1 of image is forms control)

<code> ::= <empty> | E  
<empty> specifies NVT ASCII code  
E specifies EBCDIC  
<pathname> ::= <any string recognized by the FTP Server at  
the site of the file>

The <file-id> syntax is the general RJE mechanism for specifying a particular file source or destination for input or output. If the <host-socket> form is used then direct transfer will be made by the RJE-Server to the named socket using the specified <attributes>. If the <host-file> form is used then the RJE-server will call upon its local FTP-user process to do the actual transfer. The data stream in this mode is either TELNET-like ASCII or blocked records (which may use column 1 for ASA carriage-control). Although A mode is permitted on input (column 1 is deleted) the usual mode is the default N. The output supplies carriage-control in the first character of each record ("blank" = single-space, "1" = new-page, etc.), while the optional N mode transfers the data only (as to a card punch, etc.).

The <pathname> is an arbitrary Unrecognized string which is saved by RJE-server and sent back over FTP to the FTP-server to retrieve or store the appropriate files.

INPATH or INPUT commands first store the specified <file-id> if one is supplied, and then the INPUT command initiates input. The INPATH name may be used to specify a file-id for later input and the INPUT command without file-id will cause input to initiate over a previously specified file-id. An INPUT "crlf" command with no previous <file-id> specified is illegal.

ABORT

ABORT

This command aborts any input retrieval in progress, discards already received records, and closes the retrieval connection. Note: ABORT with parameters is an Output Transmission control (see below).

OUTUSER/OUTPASS

OUTUSER = <user-id>  
OUTPASS = <password>

The specified user-id and password will be sent in the File Transfer request to send the output file(s). These parameters are not used by the Server in any other way. If this command does not appear, then the USER/PASS parameters are used.

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

## OUT

```
OUT <out-file> = <disp>

<out-file> ::= <empty> | <job-file-id>
    <empty> implies the primary print file of the job
<job-file-id> ::= <string representing a specific output file
    from the job as recognized by the Server>
<disp> ::= <empty><file-id> | (H) | (S)<file-id>|(D)
    <empty> specifies Transmit then discard
    (H) specifies Hold-only, do not transmit
    (S) specifies Transmit and Save
    (D) specifies discard without transmitting
```

NOTE: Parentheses are part of the above elements.

<file-id> ::= (same as for INPUT command)

This command specifies the disposition of output file(s) produced by the job. Unspecified files will be Hold-only by default. The OUTUSER, OUTPASS, and OUT commands must be specified before the INPUT command to be effective. These commands will affect any following jobs submitted by this USER over this RJE-TELNET connection. A particular job may override these commands by NET control cards on the front of the input file.

Once output disposition is specified by this OUT command or by a NET OUT card, the information is kept with the job until final output disposition, and is modifiable by the CHANGE command.

On occasion, the server may find that the destination for the output is "busy" (i.e., RFC to either Server-FTP or specified socket is refused), or that the host which should receive the output is dead. In these cases, the server should wait several minutes and then try to transmit again.

## OUTPUT RE-ROUTE

```
CHANGE <job-id><blank><out-file> = <disp>
```

This command changes the output disposition supplied with the job at submission. The <job-id> is assumed recognizable by the RJE-server, who may verify if this USER is authorized to modify the specified job. After the job is identified, the other information has the same syntax and semantics as the original OUT command. CHANGE command may be specified for a job-file-id which was not mentioned at submission time and has the same effect as an original OUT command.

## OUTPUT CONTROLS DURING TRANSMISSION

```
<command><blank><count><blank><what>
<command> ::= RESTART | RECOVER | BACK | SKIP | ABORT | HOLD
```

These commands specify (respectively):

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

Restart the transmission (new RFC, etc.)  
Recover restarts transmission from last FTP  
Restart-marker-reply  
(see FTP).  
Back up the output "count" blocks  
Skip the output forward "count" blocks  
Abort the output, discarding it  
Abort the output, but Hold it

```
<count> ::= <empty> | <integer>
<empty> implies 1 where defined
<what> ::= @<file-id> | <job-id><job-file-id>
<disp> ::= (same as for OUT command)
<file-id> ::= (same as for INPUT command)
<integer> ::= (same as for INPUT command)
<job-id> ::= <server recognized job identifier which was supplied
               at INP completion by the server>

<job-file-id> ::= <server recognized file identifier or if missing
                  then the prime printer output of the specified
                  job>
```

This collection of commands will modify the transmission of output in progress or recently aborted. If output transmission is cut-off before completion, then the RJE-server will either try to resend the entire file if the file's <disp> was Transmit-and-discard or will Hold the file for further User control if the <disp> was (S) transmit-and-Save. Either during transmission, during the Save part of a transmit-and-Save, or for a Hold-only file, the above commands may be used to control the transmission. The @<file-id> form of <what> is permitted only if transmission is actually in progress.

If the file's state is inconsistent with the command, then the command is illegal and ignored with reply.

#### STATUS

```
STATUS <job-id>
STATUS <job-id><blank><job-file-id>
```

These commands request the status of the RJE-server, a particular job, or the transmission of an output or input file, respectively. The information content of the Status reply is site dependent.

#### CANCEL/ALTER

```
CANCEL <job-id>
ALTER <job-id><blank><site dependent options>
```

These commands change the course of a submitted job. CANCEL specifies that the job is to be immediately terminated and any output discarded. ALTER provides for system dependent options such as changing job priority, process limits, Terminate without Cancel, etc.

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

OP

OP (any string)

The specified string is to be displayed to the Server site operator when any following job is initiated from the batch queue of the Server. This command usually appears in the input file as a NET OP control card, but may be a TELNET command. It is cancelled as an all-jobs command by an OP "crlf" command (no text supplied).

#### RJE CONTROL CARDS IN THE INPUT FILE

Certain RJE commands may be specified by control cards in the front of the input file. If these controls appear, they take precedence over the same command given thru the RJE-TELNET connection and affect only this specific job. All these RJE control cards must appear as the first records of the job's input-file. They all contain the control word NET in columns 1 through 3. Scanning for these controls stops when the first card without NET in col 1-3 is encountered.

The control commands appear in individual records and are terminated by the end-of-record (usually an 80 column card-image). Continuation is permitted onto the next record by the appearance of NET+ in columns 1-4 of the next record. Column 5 of the next record immediately follows the last character of the previous record.

```
NET OUTUSER = <user-id>
NET OUTPASS = <password>
NET OUT <out-file> = <disp>
NET OP <any string>
```

See the corresponding TELNET command for details. One option permitted by the NET OUTUSER and NET OUT controls not possible from the TELNET connection is specification of different OUTUSERs for different OUTS, since the TELNET stored and supplies only an initial OUTUSER, but the controls may change OUTUSERs before each OUT control is encountered.

#### RJE USE OF FILE TRANSFER PROTOCOL

Most non-TIP files will be transferred to or from the RJE-server through the FTP process. RJE-server will call upon its local FTP-user supplying the Host, File-pathname, User-id, Password, and Mode of the desired transfer. FTP-user will then connect to its FTP-server counterpart in the specified host and set up a transfer path. Data will then flow through the RJE-FTP interface in the Server, over the Network, from/to the foreign FTP-server and then from/to the specified File-pathname in the foreign host's file storage space. On output files, the file-pathname may be recognized by the foreign host as directions to a printer or the file may simply be stored; a User-RJE-process can supply an output <file-id> by default which is recognized by its own Server-FTP as routing to a printer.

Although many specifics of the RJE-Server/User-FTP interface are going to be site dependent, there are several FTP options which will be used in a standard way by RJE-Servers:

1. A new FTP connection will be initiated for each file to be transferred. The connection will be opened with the RJE User supplied User-id (OUTUSER or INUSER) and Password.

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

2. The data bytesize will be 8 bits.
3. The FTP Type, Structure, and Mode parameters are determined by the RJE transfer direction (I/O), and the <transmission> and <code> options supplied by the User:

I/O	<TRANS>	<CODE>	FTP-TYPE	FTP-STRUCTURE	FTP-MODE
I*	N	-	A	R	B
I	N	E	E	R	B
I	T	-	A	F	S
I	T	E	E	F	S
I	A	-	P	R	B
I	A	E	F	R	B

I/O	<TRANS>	<CODE>	FTP-TYPE	FTP-STRUCTURE	FTP-MODE
O*	A	-	P	R	B
O	A	E	F	R	B
O	N	-	A	R	B
O	N	E	E	R	B
O	T	-	A	F	S
O	T	E	E	F	S

(\*indicates default)

4. The service commands used will be Retrieve for input and Append (with create) for output. The FTP pathname will be the <pathname> supplied by the RJE User.
5. On output in B form, the User-FTP at the RJE-Server site will send Restart-markers at periodic intervals (like every 100 lines, or so), and will remember the latest Restart-marker-reply with the file. If the file transfer is not completed and the <disp> is (S) then the file will be held pending User intervention. The User may then use the RECOVER command to cause a FTP restart at the last remembered Restart-marker-reply.
6. The FTP Abort command will be used for the RJE ABORT and CANCEL commands.
7. For transfers where the FTP-MODE is defined as B, the user FTP may optionally attempt to use H mode.

The specific form of the FTP commands used by an RJE-Server site, and the order in which they are used will not be specified in this protocol.

Errors encountered by FTP fall into three categories: a) access errors or no storage space error; b) command format errors; and c) transfer failure errors. Since the commands are created by the RJE-Server process, an error is a programming problem and should be logged for attention and the situation handled as safely as possible. Transmission failure or access failure on input cause an effective ABORT and user notification. Transmission failure on output causes RESTART or Save depending on <disp> (see OUT command). Access failure on output is a problem since the User may not be accessible. A status response should be queued for him, should he happen to inquire; a <disp> = (S) file should be Held; and a <disp> = <empty> transmit-and-discard file should be temporarily held and then discarded if not claimed. "Temporarily" is understood here to mean at least several days, since particularly in

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

the case of jobs which generate voluminous output at great expense to the User, he should be given every chance to retrieve his rightful output. Servers may elect, however, to charge the User for the file-storage space occupied by the held output.

#### REPLIES OVER THE TELNET CONNECTION

Each action of the RJE-server, including entry of each TELNET command, is noted over the TELNET connection to the User. These RJE-server replies are formatted for Human or Process interpretation. They consist of a leading 3-digit numeric code followed by a blank followed by a text explanation of the message. The numeric codes are assigned by groups for future expansion to hopefully cover other protocols besides RJE (like FTP). The numeric code is designed for ease of interpretation by processes. The three digits of the code are interpreted as follows:

The first digit specified the "type" of response indicated:

000

These "replies" are purely informative, and are issued voluntarily by the Server to inform a User of some state of the server's system.

100

Replies to a specific status inquiry. These replies serve as both information and as acknowledgment of the status request.

200

Positive acknowledgment of some previous command/request. The reply 200 is a generalized "ok" for commands which require no other comment. Other 2xx replies are specified for specific successful actions.

300

Incomplete information supplied so far. No major problem, but activity cannot proceed with the input specified.

400

Unsuccessful reply. A request was correctly specified, but could not be correctly completed. Further attempts will require User commands.

500

Incorrect or illegal command. The command or its parameters were invalid or incomplete from a syntactic view, or the command is inconsistent with a previous command. The command in question has been totally ignored.

600-900

Reserved for expansion

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

The second digit specifies the general subject to which the response refers:

x00-x29

General purpose replies, not assignable to other subjects.

x30

Primary access. These replies refer to the attempt to "log-on" to a Server service (RJE, FTP, etc.).

x40

Secondary access. The primary Server is commenting on its ability to access a secondary service (RJE must log-on to a remote FTP service).

x50

FTP results.

x60

RJE results.

x70-x99

Reserved for expansion.

The final digit specifies a particular message type. Since the code is designed for an automaton process to interpret, it is not necessary for every variation of a reply to have a unique number, only that the basic meaning have a unique number. The text of a reply can explain the specific reason for the reply to a human User.

Each TELNET line (ended by "crlf") from the Server is intended to be a complete reply message. If it is necessary to continue the text of a reply onto following lines, then those continuation replies contain the special reply code of three blanks.

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

The assigned reply codes relating to RJE are:

- 000 General information message (time of day, etc.)
- 030 Server availability information
- 050 FTP commentary or user information
- 060 RJE or Batch system commentary or information
- 100 System status reply
- 150 File status reply
- 151 Directory listing reply
- 160 RJE system general status reply
- 161 RJE job status reply
- 200 Last command received ok
- 201 An ABORT has terminated activity, as requested
- 202 ABORT request ignored, no activity in progress
- 203 The requested Transmission Control has taken effect
- 204 A REINIT command has been executed, as requested
- 230 Log-on completed
- 231 Log-off completed, goodbye.
- 232 Log-off noted, will complete when transfer done
- 240 File transfer has started
- 250 FTP File transfer started ok
  - Text is: MARK yyyy = mmmm
    - where yyyy is data stream marker value (yours)
    - and mmmm is receiver's equivalent mark (mine)
- 251 FTP Restart-marker-reply
- 252 FTP transfer completed ok
- 253 Rename completed
- 254 Delete completed
- 260 Job <job-id> accepted for processing
- 261 Job <job-id> completed, awaiting output transfer
- 262 Job <job-id> Cancelled as requested
- 263 Job <job-id> Altered as requested to state <status>
- 264 Job <job-id>,<job-file-id> transmission in progress
- 300 Connection greeting message, awaiting input
- 301 Current command not completed (may be sent after suitable delay, if not "crlf")
- 330 Enter password (may be sent with hide-your-input mode)
- 360 INPUT has never specified an INPATH
- 400 This service is not implemented
- 401 This service is not accepting log-on now, goodbye.
- 430 Log-on time or tries exceeded, goodbye.
- 431 Log-on unsuccessful, user and/or password invalid
- 432 User not valid for this service
- 434 Log-out forced by operator action, please phone site
- 435 Log-out forced by system problem
- 436 Service shutting down, goodbye
- 440 RJE could not log-on to remote FTP for input transfer
- 441 RJE could not access the specified input file thru FTP
- 442 RJE could not establish <host-socket> input connection
- 443 RJE could not log-on to remote FTP for output delivery
- 444 RJE could not access file space given for output
- 445 RJE could not establish <host-socket> output connection
- 450 FTP: The named file does not exist (or access denied)
- 451 FTP: The named file space not accessible by YOU
- 452 FTP: Transfer not completed, data connection closed
- 453 FTP: Transfer not completed, insufficient storage space
- 460 Job input not completed, ABORT performed
- 461 Job format not acceptable for processing, Cancelled
- 462 Job previously accepted has mysteriously been lost
- 463 Job previously accepted did not complete

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

- 464 Job-id referenced by STATUS, CANCEL, ALTER, CHANGE, or Transmission Control is not known (or access denied)
- 465 Request Alteration is not permitted for the specified job
- 466 Un-deliverable, un-claimed output for <job-id> discarded
- 467 Requested REINIT not accomplished
- 500 Last command line completely unrecognized
- 501 Syntax of the last command is incorrect
- 502 Last command incomplete, parameters missing
- 503 Last command invalid, illegal parameter combination
- 504 Last command invalid, action not possible at this time
- 505 Last command conflicts illegally with previous command(s)
- 506 Requested action not implemented by this Server
- 507 Job <job-id> last command line completely unrecognized
- 508 Job <job-id> syntax of the last command is incorrect
- 509 Job <job-id> last command incomplete, parameters missing
- 510 Job <job-id> last command invalid, illegal parameter combination
- 511 Job <job-id> last command invalid, action impossible at this time
- 512 Job <job-id> last command conflicts illegally with previous command(s)

REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

#### SEQUENCING OF COMMANDS AND REPLIES

The communication between the User and Server is intended to be an alternating dialogue. As such, the User issues an RJE command and the Server responds with a prompt primary reply. The User should wait for this initial success or failure response before sending further commands.

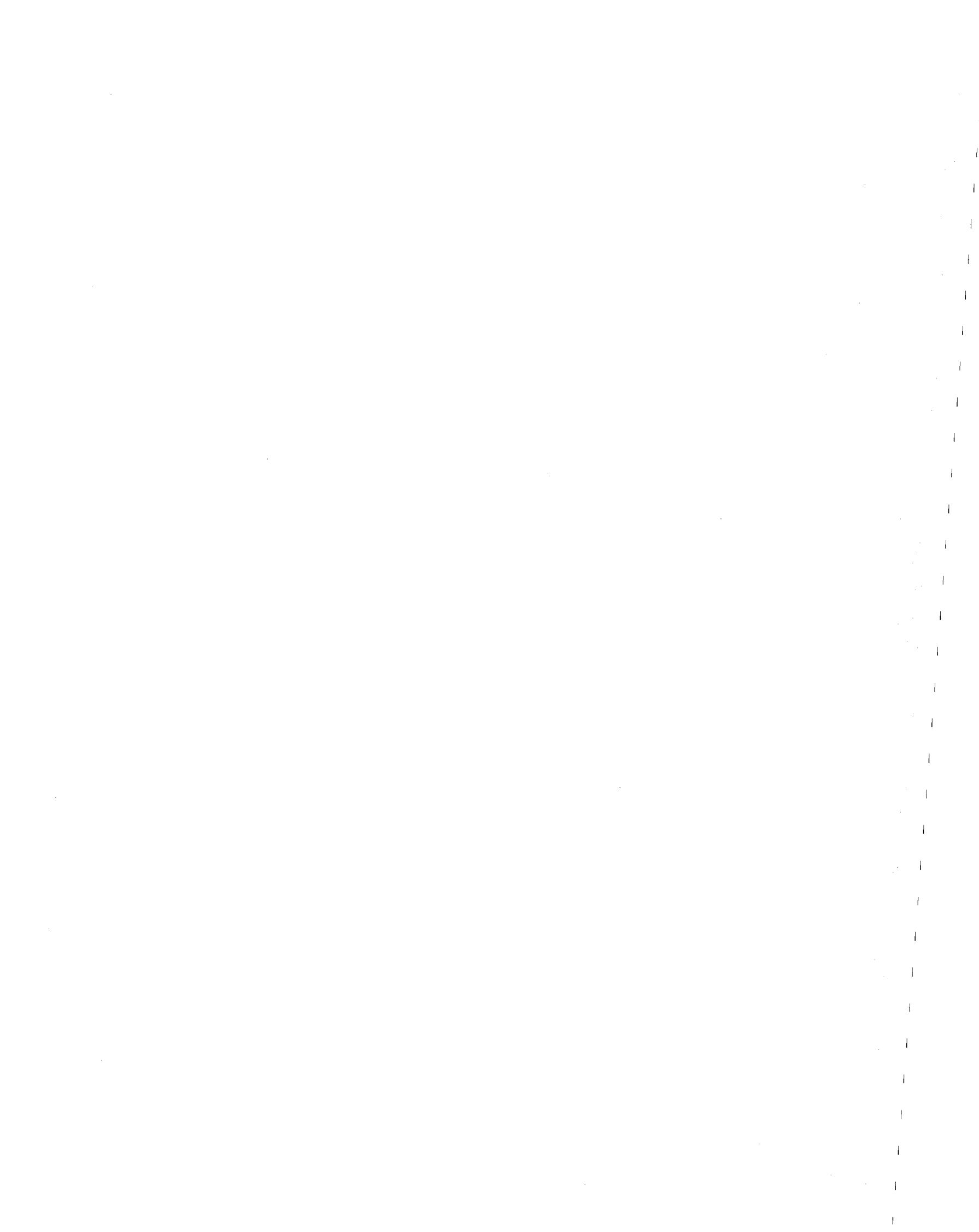
A second type of reply is sent by Server asynchronously with respect to User commands. These replies report on the progress of a job submission caused by the INPUT command and as such are secondary replies to that command.

The final class of Server "replies" are strictly informational and may arrive at any time. These "replies" are listed below as spontaneous.

#### COMMAND-REPLY CORRESPONDENCE TABLE

COMMAND	SUCCESS	FAILURE
REINIT	204	467, 500-505
USER	230, 330	430-432, 500-505
PASS	230	430-432, 500-505
BYE	231, 232	500-505
INID	200	500-505
INPASS	200	500-505
INPATH	200	500-505
INPUT	240	360, 440-442, 500-505
sec. input retrieval	260	460, 461
sec. job execution	261	462, 463
sec. output transmission	-	443-445, 466
ABORT (input)	201, 202	500-505
OUTUSER	200	500-505
OUTPASS	200	500-505
OUT	200	500-505
CHANGE	200	500-505
RESTART/RECOVER/BACK		
/SKIP/ABORT (output)/HOLD	203	464, 500-506
STATUS	1xx, 264	460-465, 500-505
CANCEL	262	464, 500-506
ALTER	263	464, 465, 500-506
OP	200	500-505
Spontaneous	0xx, 300, 301	434-436

NOTE: For commands appearing on cards, a separate set of error codes is provided (507-512). Since these error replies are "asynchronously" sent, and thus could cause some confusion if the user is in the process of submitting a new job after the present one, the error replies must identify which job has the faulty card(s).



REMOTE JOB ENTRY PROTOCOL  
RFC 407, NIC 12112 (Oct. 16, 1972)

TYPICAL RJE SCENARIOS

TIP USER WANTING HOT CARD READER TO HOSTX

1. TIP user opens TELNET connection to HOSTX socket 5
2. Commands sent over TELNET to RJE

```
USER=myself
PASS=dorwssap
OUT=H70002
INPUT=H50003
```

3. RJE-server connects to the TIP's device 5 and begins reading. When end-of-job card is recognized, the job is queued to run. The connection to the card reader is still open for more input as another job.
4. The first job finishes. A connection to the TIP's device 7 is established by RJE-server and the output is sent as an NVT stream.
5. Continue at any time with another deck at step 3.

TIP WITH JOB-AT-A-TIME CARD READER

1. thru 4. the same but User closes Reader after the deck
2. The output finishes and the printer connection closes.
3. INPUT may be typed any time after step 3 finishes and another job will be entered starting at 3.

HOSTA USER RUNS JOB AT HOSTC, INPUT FROM HOSTB

1. User TELNET connects to HOSTC socket 5 for RJE

```
USER=roundabout
PASS=aaabbcc
OUTUSER=roundab1
OUT=:E/.sysprinter
OUT puncher = (S)HOSTB:NE/my.savepunch INUSER=rounder
INPASS=x.x.x
INPUT=HOSTB:E/my.jobinput
```

2. The RJE-server has FTP retrieve the input from HOSTB using User-id of "rounder" and Password of "x.x.x" for file named "my.jobinput".
3. The job finishes. RJE-server uses FTP to send two files: the print output is sent to HOSTA in EBCDIC with ASA carriage control to file ".sysprinter" while the file known as "puncher" is sent to HOSTB in EBCDIC without carriage-control to file "my.savepunch".
4. when the outputs finish, RJE-server at HOSTC discards the print file but retains the "puncher" file.

Preceding page blank

**REMOTE JOB ENTRY PROTOCOL**  
**RFC 407, NIC 12112 (Oct. 16, 1972)**

5. The User who has signed out after job submission has gotten his output and checked his file "my.savepunch" at HOSTB. He deletes the saved copy at HOSTC by re-calling RJE at HOSTC.

USER=roundabout  
PASS=aaabbbcc  
ABORT job 123 puncher  
or  
CHANGE job 123 puncher = (D)

**NETRJS PROTOCOL**  
**RFC 740, NIC 42423 (Nov. 22, 1977)**

R. Braden (UCLA-CCN)  
RFC 740, NIC: 42423 (Nov. 22, 1977)  
Obsoletes: 189, 599

**NETRJS PROTOCOL**

**INTRODUCTION**

NETRJS, a private protocol for remote job entry service, was defined and implemented by the UCLA Campus Computing Network (CCN) for batch job submission to an IBM 360 Model 91. CCN's NETRJS server allows a remote user, or a daemon process working in behalf of a user, to access CCN's RJS ("Remote Job Service") subsystem. RJS provides remote job entry service to real remote batch (card reader/line printer) terminals over direct communications lines as well as to the ARPANET.

A batch user at a remote host needs a NETRJS user process to communicate with the NETRJS server at the batch host. An active NETRJS user process simulates a "Virtual Remote Batch Terminal", or "VRBT".

A VRBT may have virtual card readers, printers, and punches. In addition, every VRBT has a virtual remote operator console. Using a virtual card reader, a Network user can transmit a stream of card images comprising one or more batch jobs, complete with job control language ("JCL"), to the batch server host. The NETRJS server will cause these jobs to be spooled into the batch system to be executed according to their priority. NETRJS will automatically return the print and/or punch output images which are created by these jobs to the virtual printer and/or card punch at the VRBT from which the job was submitted. The batch user can wait for his output, or he can signoff and signon again later to receive it.

To initiate a NETRJS session, the user process must execute a standard ICP to a fixed socket at the server. The result is to establish a full-duplex Telnet connection for the virtual remote operator console, allowing the VRBT to signon to RJS. The virtual remote operator console can then be used to issue commands to NETRJS and to receive status, confirmation, and error messages from the server. The most important remote operator commands are summarized in Appendix D.

Different VRBT's are distinguished by 8-character terminal id's, which are assigned by the server site to individual batch users or user groups.

**CONNECTIONS AND PROTOCOLS**

The protocol uses up to five connections between the user and server processes. The operator console uses a full-duplex Telnet connection. The data transfer streams for the virtual card reader, printer, and punch each use a separate simplex connection under a data transfer protocol defined in Appendix A. This document will use the term "channel" for one of these simplex data transfer connections and will designate a connection "input" or "output" with reference to the server.

## NETRJS PROTOCOL

RFC 740, NIC 42423 (Nov. 22, 1977)

A particular data transfer channel needs to be open only while it is in use, and different channels may be used sequentially or simultaneously. CCN's NETRJS server will support simultaneous operation of a virtual card reader, a virtual printer, and a virtual punch (in addition to the operator console) on the same VRBT process. The NETRJS protocol could easily be extended to any number of simultaneously-operating virtual card readers, printers, and punches.

The NETRJS server takes a passive role in opening the data channels: the server only "listens" for an RFC from the user process. NETRJS is defined with an 8-bit byte size on all data channels.

Some implementations of NETRJS user processes are daemons, operating as background processes to submit jobs from a list of user requests; other implementations are interactive processes executed directly under terminal control by remote users. In the latter case, the VRBT process generally multiplexes the user terminal between NETRJS, i.e., acting as the remote operator console, and entering local commands to control the VRBT. Local VRBT commands allow selection of the files containing job streams to be sent to the server as well as files to receive job output from the server. Other local commands would cause the VRBT to open data transfer channels to the NETRJS server and to close these channels to free buffer space or abort transmission.

The user process has a choice of three ICP sockets, to select the character set of the VRBT -- ASCII-68, ASCII-63, or EBCDIC. The server will make the corresponding translation of the data in the card reader and printer channels. (In the CCN implementation of NETRJS, an EBCDIC VRBT will transmit and receive, without translation, "transparent" streams of 8-bit bytes, since CCN is an EBCDIC installation). The punch stream will always be transparent, outputting "binary decks" of 80-byte records untranslated. The operator console connections always use Network ASCII, as defined by the Telnet protocol.

The NETRJS protocol provides data compression, replacing repeated blanks or other characters by repeat counts. However, when the terminal id is assigned, a particular network VRBT may be specified to use no data compression. In this case, NETRJS will simply truncate trailing blanks and send records in a simple "op code-length-data" form, called "truncated format" (see Appendix A).

### STARTING AND TERMINATING A SESSION

The remote user establishes a connection to the NETRJS server by executing an ICP to the contact socket 71 (decimal) for EBCDIC, socket 73 (decimal) for ASCII-68, or to socket 75 (decimal) for ASCII-63. A successful ICP results in a pair of connections which are in fact the NETRJS operator console connections. NETRJS will send a READY message over the operator output connection.

The user (process) must now enter a valid NETRJS signon command ("SIGNON terminal-id") through the virtual remote operator console. RJS will normally acknowledge signon with a console message; however, if there is no available NETRJS server port, NETRJS will indicate refusal by closing both operator connections. If the user fails to enter a valid signon within 3 minutes, NETRJS will close the operator connections. If the VRBT attempts to open data transfer channels before the signon command is accepted, the data transfer channels will be refused with an error message to the VRBT operator console.

NETRJS PROTOCOL  
RFC 740, NIC 42423 (Nov. 22, 1977)

Suppose that S is the even number sent in the ICP; then the NETRJS connections have sockets at the server with fixed relation to S, as shown in the following table:

CHANNEL	SERVER SOCKET	USER SOCKET
Remote Operator Console Input	S	U + 3 Telnet
Remote Operator Console Output	S + 1	U + 2 Telnet
Data Transfer - Card Reader #1	S + 2	any odd number
Data Transfer - Printer #1	S + 3	any even number
Data Transfer - Punch #1	S + 5	any even number

Once the VRBT has issued a valid signon, it can open data transfer channels and initiate input and output operations as explained in the following sections. To terminate the session, the VRBT may close all connections. Alternatively, it may enter a SIGNOFF command through the virtual remote operator console. Receiving a SIGNOFF, NETRJS will wait until the current job output streams are complete and then itself terminate the session by closing all connections.

#### INPUT OPERATIONS

A job stream for submission to the NETRJS server is a series of logical records, each of which is a card image of at most 80 characters. The user can submit a "stack" of successive jobs through the card reader channel with no end-of-job indication between jobs; NETRJS is able to parse the JCL sufficiently to recognize the beginning of each job.

To submit a batch job or stack of jobs for execution, the user process must first open the card reader channel by issuing an Init for foreign socket S+2 and the appropriate local socket. NETRJS, which is listening on socket S+2, will return an RTS command to open the channel. When the channel is open, the user can begin sending his job stream using the protocol defined in Appendix A. For each job successfully spooled, NETRJS will send a confirming message to the remote operator console.

At the end of the job stack, the user process must send an End-of-Data transaction to initiate processing of the last job. NETRJS will then close the channel (to avoid holding buffer space unnecessarily). At any time during the session, the user process can re-open the card reader channel and transmit another job stack. It can also terminate the session and signon later to get the output.

If the user process leaves the channel open for 5 minutes without sending any bits, the server will abort (close) the channel. The user process can abort the card reader channel at any time by closing the channel; NETRJS will then discard the last partially spooled job. If NETRJS finds an error (e.g., transaction sequence number error or a dropped bit), it will abort the channel by closing the channel prematurely, and also inform the user process that the job was discarded (thus solving the race condition between End-of-Data and aborting). The user process should retransmit only those jobs in the stack that have not been completely spooled.

If the user's process, NCP, or host, or the Network itself fails during input, RJS will discard the job being transmitted. A message informing the user that this job was discarded will be generated and sent to him the next time he signs on. On the other hand, those jobs whose receipt have been acknowledged on the operator's console will not be affected by the failure, but will be executed by the server.

## NETRJS PROTOCOL

RFC 740, NIC 42423 (Nov. 22, 1977)

### OUTPUT OPERATIONS

The VRBT may wait to set up a virtual printer or punch and open its channel until a STATUS message from NETRJS indicates output is ready; or it may leave the output channel(s) open during the entire session, ready to receive output whenever it becomes available. The VRBT can also control which one of several available jobs is to be returned by entering appropriate operator commands.

To be prepared to receive printer (or punch) output from its jobs, the VRBT issues an Init for foreign socket S+3 or S+5 for printer or punch output, respectively. NETRJS is listening on these sockets and should immediately return an STR. However, it is possible that because of a buffer shortage, NETRJS will refuse the connection by returning a CLS; in this case, try again later.

When NETRJS has job output for a particular virtual terminal and a corresponding open output channel, it will send the output as a series of logical records using the protocol in Appendix A. The first record will consist of the job name (8 characters) followed by a comma and then the ID string from the JOB card, if any. In the printer stream, the first column of each record after the first will be an ASA carriage control character (see Appendix C). A virtual printer in NETRJS has 254 columns, exclusive of carriage control; NETRJS will send up to 255 characters of a logical record it finds in a SYSOUT data set. If the user wishes to reject or fold records longer than some smaller record size, he can do so in his VRBT process.

NETRJS will send an End-of-Data transaction and then close an output channel at the end of the output for each complete batch job; the remote site must then send a new RFC to start output for another job. This gives the remote site a chance to allocate a new file for each job without breaking the output within a job.

If the batch user wants to cancel (or backspace or defer) the output of a particular job, he can enter appropriate NETRJS commands on the operator input channel (see Appendix D).

If NETRJS encounters a permanent I/O error in reading the disk data set, it will notify the user via his console, skip forward to the next set of system messages or SYSOUT data set in the same job, and continue. If the user process stops accepting bits for 5 minutes, the server will abort the channel. In any case, the user will receive notification of termination of output data transfer for each job via a remote console message.

If the user detects an error in the stream, he can issue a Backspace (BSP) command from his console to repeat the last "page" of output, or a Restart (RST) command to repeat from the last SYSOUT data set or the beginning of the job, or he can abort the channel by closing his socket. If he aborts the channel, NETRJS will simulate a Backspace command, and when the user re-opens the channel the job will begin transmission again from an earlier point in the same data set. This is true even if the user terminates the current session first and reopens the channel in a later session; RJS saves the state of every incomplete output stream. However, before re-opening the channel he can defer this job for later output, restart it at the beginning, or cancel its output (see Appendix D). Note that aborting the channel is only effective if NETRJS has not yet sent the End-of-Data transaction.

If the user's process, NCP, or host or the Network itself fails during an output operation, NETRJS will act as if the channel had been aborted and the user signed off. NETRJS will

**NETRJS PROTOCOL  
RFC 740, NIC 42423 (Nov. 22, 1977)**

discard the output of a job only after receiving the RFNM from the last data transfer message (containing an End-of-Data). In no case should a NETRJS user lose output from a batch job.



NETRJS PROTOCOL  
RFC 740, NIC 42423 (Nov. 22, 1977)

APPENDIX A  
DATA TRANSFER PROTOCOL IN NETRJS

**1. INTRODUCTION**

The records in the data transfer channels (for virtual card reader, printer, and punch) are generally grouped into transactions preceded by headers. The transaction header includes a sequence number and the length of the transaction. Network byte size must be 8 bits in these data streams.

A transaction is the unit of buffering within the server software, and is limited to 880 8-bit bytes. Transactions can be as short as one record; however, those sites which are concerned with efficiency should send transactions as close as possible to the 880 byte limit.

There is no necessary connection between physical message boundaries and transactions ("logical messages"); the NCP can break a transaction arbitrarily into physical messages. The CCN server starts each transaction at the beginning of a new physical message, but this is not a requirement of the protocol.

Each logical record within a transaction begins with an "op code" byte which contains the channel identification, so its value is unique to each channel but constant within a channel. This choice provides the receiver with a convenient way to verify bit-synchronization, and it also allows an extension in the future to true "multi-leaving" (i.e., multiplexing all channels within one connection in each direction).

The only provisions for transmission error detection in the current NETRJS protocol are (1) the "op code" byte to verify bit synchronization and (2) the transaction sequence number. Under the NETRJS protocol, a data transfer error must abort the entire transmission; there is no provision for restart.

**2. META-NOTATION**

The following description of the NETRJS data transfer protocol uses a formal notation derived from that proposed in RFC 31 by Bobrow and Sutherland. The notation consists of a series of productions for bit string variables. Each variable name which represents a fixed length field is followed by the length in bits (e.g., SEQNUMB(16)). Numbers enclosed in quotes are decimal, unless qualified by a leading X meaning hex. Since each hex digit is 4 bits, the length is not shown explicitly in hex numbers. For example, '255'(8) and X'FF' both represent a string of 8 one bits.

Preceding page blank

## NETRJS PROTOCOL

RFC 740, NIC 42423 (Nov. 22, 1977)

The meta-syntactic operators are:

- | :alternative string
- [ ] :optional string
- ( ) :grouping
- + :catenation of bit strings

The numerical value of a bit string (interpreted as an integer) is symbolized by a lower case identifier preceding the string expression and separated by a colon. For example, in "i:FIELD(8)", i symbolizes the numeric value of the 8 bit string FIELD.

Finally, we use Bobrow and Sutherland's symbolism for iteration of a sub-string: (STRING-EXPRESSION = n); denotes n occurrences of STRING-EXPRESSION, implicitly catenated together. Here any n greater or equal to 0 is assumed unless n is explicitly restricted.

### 3. PROTOCOL DEFINITION

STREAM ::= (TRANSACTION = n) + [END-OF-DATA]

That is, STREAM, the entire sequence of data on a particular open channel, is a sequence of n TRANSACTIONS followed by an END-OF-DATA marker (omitted if the sender aborts the channel).

TRANSACTION ::= THEAD(72) + (RECORD = r) + ('0'(1) = f)

That is, a transaction consists of a 72 bit header, r records, and f filler bits; it may not exceed 880\*8 bits.

THEAD ::= X'FF'+f:FILLER(8)+SEQNUMB(16)+LENGTH(32)+X'00'

Transactions are to be consecutively numbered in the SEQNUMB field, starting with 0 in the first transaction after the channel is (re-) opened. The 32 bit LENGTH field gives the total length in bits of the r RECORD's which follow. For convenience, the using site may add f additional filler bits at the end of the transaction to reach a convenient word boundary on his machine; the value f is transmitted in the FILLER field of THEAD.

RECORD ::= COMPRESSED | TRUNCATED

RJS will accept intermixed RECORD's which are COMPRESSED or TRUNCATED in an input stream. RJS will send one or the other format in the printer and punch streams to a given VRBT; the choice is determined for each terminal id.

COMPRESSED ::= '2'(2) + DEVID(6) + (STRING = p) + '0'(8)

STRING ::= ('6'(3) + i:DUPCOUNT(5)) |

This form represents a string of i consecutive blanks

NETRJS PROTOCOL  
RFC 740, NIC 42423 (Nov. 22, 1977)

('7'(3) + i:DUPCOUNT(5) + TEXTBYTE(8)) |

This form represents string of i consecutive duplicates of TEXTBYTE.

('2'(2) + j:LENGTH(6) + (TEXTBYTE(8) = j))

This form represents a string of j characters.

TRUNCATED ::= '3'(2) + DEVID(6) + n:COUNT(8) + (TEXTBYTE(8)=n)

DEVID(6) ::= DEVNO(3) + t:DEVTYPE(3)

DEVID identifies a particular virtual device, i.e., it identifies a channel. DEVTYPE specifies the type of device, as follows:

- t = 1: Output to remote operator console
- 2: Input from remote operator console
- 3: Input from card reader
- 4: Output to printer
- 5: Output to card punch
- 6: Unused
- 7: Unused

DEVNO identifies the particular device of type t at this remote site; at present only DEVNO = 0 is possible.

END-OF-DATA ::= X'FE'

Signals end of job (output) or job stack (input).



**NETRJS PROTOCOL**  
**RFC 740, NIC 42423 (Nov. 22, 1977)**

APPENDIX B

TELNET FOR VRBT OPERATOR CONSOLE

The remote operator console connections use the ASCII Telnet protocol. Specifically:

1. The following one-to-one character mappings are used for the three EBCDIC graphics not in ASCII:

ASCII in TELNET	NETRJS
broken vertical bar	solid vertical bar
tilde	not sign
back slash	cent sign.

2. Telnet controls are ignored.
3. An operator console input line which exceeds 133 characters (exclusive of CR LF) is truncated by NETRJS.
4. NETRJS accepts BS (Control-H) to delete a character and CAN (Control-X) to delete the current line. The sequence CR LF terminates each input and output line. HT (Control-I) is translated to a single space. An ETX (Control-C) terminates (aborts) the session. All other ASCII control characters are ignored.
5. NETRJS translates the six ASCII graphics with no equivalent in EBCDIC into the character question mark ("?") on input.



**NETRJS PROTOCOL**  
**RFC 740, NIC 42423 (Nov. 22, 1977)**

APPENDIX C

CARRIAGE CONTROL

The carriage control characters sent in a printer channel by NETRJS conform to IBM's extended USASI code, defined by the following table:

CODE	ACTION BEFORE WRITING RECORD
Blank	Space one line before printing
0	Space two lines before printing
-	Space three lines before printing
+	Suppress space before printing
1	Skip to channel 1
2	Skip to channel 2
3	Skip to channel 3
4	Skip to channel 4
5	Skip to channel 5
6	Skip to channel 6
7	Skip to channel 7
8	Skip to channel 8
9	Skip to channel 9
A	Skip to channel 10
B	Skip to channel 11
C	Skip to channel 12



NETRJS PROTOCOL  
RFC 740, NIC 42423 (Nov. 22, 1977)

APPENDIX D

NETWORK/RJS COMMAND SUMMARY

This section presents an overview of the RJS Operator Commands, for the complete form and parameter specifications please see references 2 and 3.

**TERMINAL CONTROL AND INFORMATION COMMANDS**

SIGNON	First command of a session; identifies VRBT by giving its terminal id.
SIGNOFF	Last command of a session; RJS waits for any data transfer in progress to complete and then closes all connections.
STATUS	Outputs on the remote operator console a complete list, or a summary, of all jobs in the system for this VRBT, with an indication of their processing status in the batch host.
ALERT	Outputs on the remote operator console an "Alert" message, if any, from the computer operator. The Alert message is also automatically sent when the user does a SIGNON, or whenever the message changes.
MSG	Sends a message to the computer operator or to any other RJS terminal (real or virtual). A message from the computer operator or another RJS terminal will automatically appear on the remote operator console.

**JOB CONTROL AND ROUTING COMMANDS**

Under CCN's job management system, the default destination for output is the input source. Thus, a job submitted under a given VRBT will be returned to that VRBT (i.e., the same terminal id), unless the user's JCL overrides the default destination.

RJS places print and punch output destined for a particular remote terminal into either an Active Queue or a Deferred Queue. When the user opens his print or punch output channel, RJS immediately starts sending job output from the Active Queue, and continues until this queue is empty. Job output in the Deferred Queue, on the other hand, must be called for by job name, (via a RESET command from the remote operator) before RJS will send it. The Active/Deferred choice for output from a job is determined by the deferral status of the VRBT when the job is entered; the deferral status, which is set to the Active option when the user signs on, may be changed by the SET command.

SET	Allows the remote user to change certain properties of his VRBT for the duration of the current session;
	(a) May change the default output destination to be another (real or virtual) RJS terminal or the central facility.
	(b) May change the deferral status of the VRBT.
DEFER	Moves the print and punch output for a specified job or set of jobs from the Active Queue to the Deferred Queue. If the job's output is in the process of being transmitted over a channel, RJS aborts the channel and

Preceding page blank

## NETRJS PROTOCOL

RFC 740, NIC 42423 (Nov. 22, 1977)

saves the current output location before moving the job to the Deferred Queue. A subsequent RESET command will return it to the Active Queue with an implied Backspace (BSP).

RESET	Moves specified job(s) from Deferred to Active Queue so they may be sent to user. A specific list of job names or all jobs can be moved with one RESET command.
ROUTE	Re-routes output of specified jobs (or all jobs) waiting in the Active and Deferred Queues for the VRBT. The new destination may be any other RJS terminal or the central facility.
ABORT	Cancels a job which was successfully submitted and awaiting execution or is currently executing.

### OUTPUT STREAM CONTROL COMMANDS

BSP (BACKSPACE)	"Backspaces" output stream within current sysout data set. Actual amount backspaced depends upon sysout blocking but is roughly equivalent to a page on the line printer.
CAN (CANCEL)	(a) On an output channel, CAN causes the rest of the output in the sysout data set currently being transmitted to be omitted. Alternatively, may omit the rest of the sysout data sets for the job currently being transmitted; however, the remaining system and accounting messages will be sent.  (b) On an input channel, CAN causes RJS to ignore the job currently being read. However, the channel is not aborted as a result, and RJS will continue reading in jobs on the channel.  (c) CAN can delete all sysout data sets for specified job(s) waiting in Active or Deferred Queue.
RST (RESTART)	(a) Restarts a specified output stream at the beginning of the current sysout data set or, optionally, at the beginning of the job.  (b) Marks as restarted specified job(s) whose transmission was earlier interrupted by system failure or user action (e.g., DEFER command or aborting the channel). When RJS transmits these jobs again it will start at the beginning of the partially transmitted sysout data set or, optionally, at the beginning of the job. This function may be applied to jobs in either the Active or the Deferred Queue; however, if the job was in the Deferred Queue then RST also moves it to the Active Queue. If the job was never transmitted, RST has no effect other than this queue movement.
REPEAT	Sends additional copies of the output of specified jobs.
EAM	Echoes the card reader stream back in the printer and/or punch stream.

**NETRJS PROTOCOL**  
**RFC 740, NIC 42423 (Nov. 22, 1977)**

APPENDIX E

NETRJS TERMINAL OPTIONS

When a new NETRJS virtual terminal is defined, certain options are available; these options are listed below.

1. Truncated/Compressed Data Format

A VRBT may use either the truncated data format (default) or the compressed format for printer and punch output. See Reference 9 for discussion of the virtues of compression.

2. Automatic Coldstart Job Resubmission

If "R" (Restart) is specified in the accounting field on the JOB card and if this option is chosen, RJS will automatically resubmit the job from the beginning if the server operating system should be "coldstarted" before all output from the job is returned. Otherwise, the job will be lost and must be resubmitted from the remote terminal in case of a coldstart.

3. Automatic Output RESTART

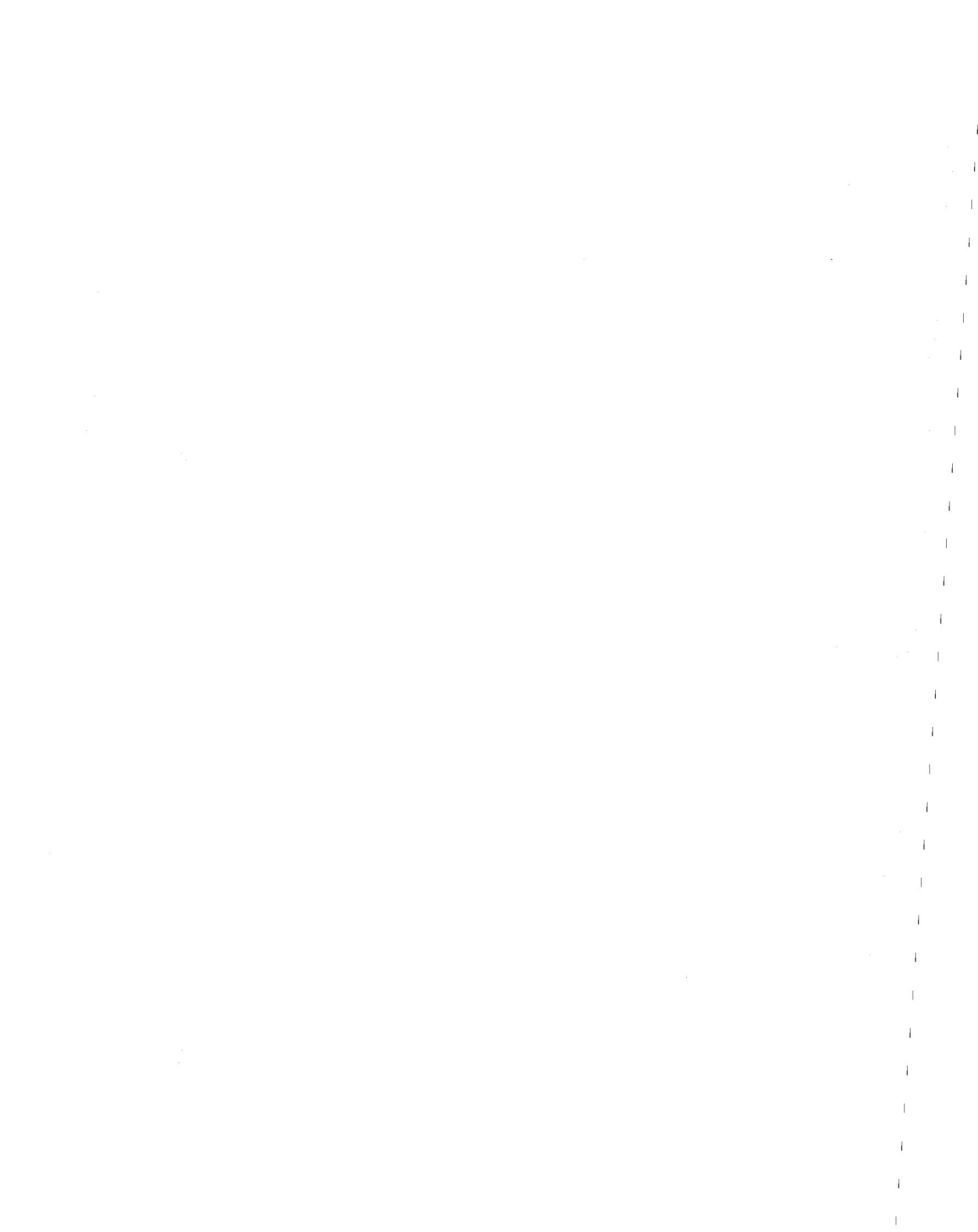
With this option, transmission of printer output which is interrupted by a broken connection always starts over at the beginning. Without this option, the output is backspaced approximately one page when restarted, unless the user forces the output to start over from the beginning with a RESTART command when the printer channel is re-opened and before printing begins.

4. Password Protection

This option allows a password to be supplied when a terminal is signed on, preventing unauthorized use of the terminal ID.

5. Suppression of Punch Separator and Large Letters.

This option suppresses both separator cards which RJS normally puts in front of each punched output deck, and separator pages on printed output containing the job name in large block letters. These separators are an operational aid when the output is directed to a real printer or punch, but generally undesirable for an ARPA user who is saving the output in a file for on-line examination.



NETRJS PROTOCOL  
RFC 740, NIC 42423 (Nov. 22, 1977)

APPENDIX F  
CHARACTER TRANSLATION BY CCN SERVER

A VRBT declares its character set for job input and output by the initial connection socket it chooses. A VRBT can have the ASCII-68, the ASCII-63, or the EBCDIC character set. The ASCII-63 character mapping was added to NETRJS at the request of users whose terminals are equipped with keyboards like those found on the model 33 Teletype.

Since CCN operates an EBCDIC machine, its NETRJS server translates ASCII input to EBCDIC and translates printer output back to ASCII. The details of this translation are described in the following.

For ASCII-68, the following rules are used:

1. There is one-to-one mapping between the three ASCII characters broken vertical bar, tilde, and back slash, which are not in EBCDIC, and the three EBCDIC characters vertical bar, not sign, and cent sign (respectively), which are not in ASCII.
2. The other six ASCII graphics not in EBCDIC are translated on input to unused EBCDIC codes, shown in the table below.
3. The ASCII control DC4 is mapped to and from the EBCDIC control TM.
4. The other EBCDIC characters not in ASCII are mapped in the printer stream into the ASCII question mark.

For ASCII-63, the same rules are used except that the ASCII-63 codes X'60' and X'7B' – X'7E' are mapped as in the following table.

EBCDIC	ASCII-68 VRBT	ASCII-63 VRBT			
vertical bar	X'4F'	vertical bar	X'7C'	open bracket	X'5B'
not sign	X'5F'	tilde	X'7E'	close bracket	X'5D'
cent sign	X'4A'	back slash	X'5C'	back slash	X'5C'
underscore	X'6D'	underscore	X'5F'	left arrow	X'5F'
	X'71'	up arrow	X'5E'	up arrow	X'5E'
open bracket	X'AD'	open bracket	X'5B'	:	X'7C'
close bracket	X'BD'	close bracket	X'5D'	:	X'7E'
.	X'8B'	open brace	X'7B'	:	X'7B'
.	X'9B'	close brace	X'7D'	:	X'7D'
.	X'79'	accent	X'60'	:	X'60'

Preceding page blank



**NETRJS PROTOCOL**  
**RFC 740, NIC 42423 (Nov. 22, 1977)**

APPENDIX G  
REFERENCES

1. Interim NETRJS Specifications, R. T. Braden. RFC 189, NIC 7133, July 15, 1971.

This was the basic system programmer's definition document. The proposed changes mentioned on the first page of RFC 189 were never implemented, since the DTP then in vogue became obsolete.

2. NETRJS Remote Operator Commands, R. T. Braden. NIC 7182, August 9, 1971

This document together with References 3 and 8 define the remote operator (i.e. user) command language for NETRJS, and form the basic user documentation for NETRJS at CCN.

3. Implementation of a Remote Job Service, V. Martin and T. W. Springer. NIC 7183, July, 1971. Operator Commands, R. T. Braden. NIC 7182, August 9, 1971

4. Remote Job Entry to CCN via UCLA Sigma 7; A scenario, UCLA/CCN. NIC 7748, November 15, 1971.

This document described the first NETRJS user implementation available on a server host. This program is no longer of general interest.

5. Using Network Remote Job Entry, E. F. Harslem. RFC 307, NIC 9258, February 24, 1972.

This document is out of date, but describes generally the Tenex NETRJS user process RJS.

6. EBCDIC/ASCII Mapping for Network RJS, R. T. Braden. RFC 338, NIC 9931, May 17, 1972.

The ASCII-63 mapping described here is no longer correct, but CCN's standard ASCII-68/EBCDIC mapping is described correctly. This information is accurately described in Appendix F of the current document.

7. NETRJT--Remote Job Service Protocol for TIP's, R. T. Braden. RFC 283, NIC 38165, December 20, 1971.

This was an attempt to define an rje protocol to handle TIPs. Although NETRJT was never implemented, many of its features are incorporated in the current Network standard RJE protocol.

8. CCN NETRJS Server Messages to Remote User, R. T. Braden. NIC 20268, November 26, 1973.

9. FTP Data Compression, R. T. Braden. RFC 468, NIC 14742, March 8, 1973.

10. Update on NETRJS, R. T. Braden. RFC 599, NIC 20854, December 13, 1973.

This updated reference 1, the current document combines the two.

11. Network Remote Job Entry -- NETRJS, G. Hicks. RFC 325, NIC 9632, April 6, 1972.

**NETRJS PROTOCOL**

**RFC 740, NIC 42423 (Nov. 22, 1977)**

12. CCNRJS: Remote Job Entry between Tenex and UCLA-CCN, D. Crocker. NUTS Note 22, [ISI]<DOCUMENTATION>CCNRJS.DOC, March 5, 1975.

13. Remote Job Service at UCSB, M. Krilanovich. RFC 477, NIC 14992, May 23, 1973.

420

**A Network Graphics Protocol**

**August 16, 1974**

**Robert F. Sproull  
Xerox Palo Alto Research Center**

**Elaine L. Thomas  
Massachusetts Institute of Technology -- Project MAC**



## Table of Contents

	SECTION	PAGE
I	<b>Guide to the Document</b>	<b>1</b>
II	<b>Introduction</b>	<b>2</b>
	II.1 <b>A Typical Session</b>	<b>3</b>
	II.2 <b>A Model for the Network Graphics Protocol</b>	<b>4</b>
	II.3 <b>Positioned Text.</b>	<b>11</b>
	II.4 <b>Input Facilities</b>	<b>11</b>
	II.5 <b>Inquiry</b>	<b>12</b>
	II.6 <b>UP Implementations.</b>	<b>13</b>
	II.7 <b>Summary.</b>	<b>14</b>
III	<b>The Protocol</b>	<b>15</b>
	III.1 <b>Initial Connection Protocol.</b>	<b>16</b>
	III.2 <b>Output Protocol Formats</b>	<b>18</b>
	III.3 <b>Transformed Format</b>	<b>18</b>
	III.4 <b>Segment Control</b>	<b>19</b>
	III.5 <b>Graphical Primitives</b>	<b>21</b>
	III.6 <b>Coordinate Systems</b>	<b>21</b>
	III.7 <b>Intensity.</b>	<b>21</b>
	III.8 <b>Line Type</b>	<b>22</b>
	III.9 <b>Character Display</b>	<b>22</b>
	III.10 <b>Segment Attributes.</b>	<b>23</b>
	III.11 <b>Segment Readback.</b>	<b>26</b>
	III.12 <b>Positioned Text.</b>	<b>26</b>
	III.13 <b>Input Facilities</b>	<b>28</b>



III.14	Reading the State of Input Devices . . . . .	29
III.15	Input Events . . . . .	30
III.16	Enabling Events. . . . .	32
III.17	Event Reports . . . . .	33
III.18	Inquiry . . . . .	35
III.19	Miscellaneous . . . . .	38
IV	Implementation Suggestions . . . . .	40
V	Op-Code Assignments and Options . . . . .	41
Appendix	. . . . .	44
References	. . . . .	49



## SECTION I

### *Guide to the Document*

This report describes a network graphics protocol (NGP) developed by the Network Graphics Group, a working group of ARPA network members. We believe that the design presented here should appeal to two groups:

- Those interested in device-independent graphics systems and graphics standards. Indeed, the philosophy behind the protocol design originates in principles of graphics system design.
- Those interested in using graphics via any network or communication system. Although the design was developed for the ARPA network, we believe that it has far wider applicability.

There are three major parts to the report: (1) an introductory section that gives the goals of the protocol, the general philosophy that gave rise to the particular protocol design, and definitions for a number of terms used throughout the report; (2) the details of the protocol; and (3) some suggestions to aid implementation of the protocol.

The authors of this report are by no means the only contributors to the ideas presented here. The Network Graphics Group members, too numerous to mention, have all contributed. Especially useful ideas were provided by Jim Michener, Dan Cohen, Ira Cotton, William Newman, Ken Victor and Ed Taft.

---

Preceding page blank

## SECTION II

### *Introduction*

Protocol proposals tend to contain so much detail that the overall intent and design considerations are lost in a pile of bits. This introduction attempts to describe the high-level considerations of the graphics protocol, in preparation for the mass of detail in section III.

The aim of a graphics protocol is to allow users with various different kinds of display hardware at different sites in a network to make use of common "graphics application programs." For example, a user may want to access the BIOMOD system at Rand, the On-Line System at UCSB or the NLS system at SRI with his or her display hardware.

One approach is a collection of "special-purpose protocols." Each application program would publish a description of the protocol needed to drive the program and to view the output (e.g. the UCSB system was so documented). The prospective user would then write a program at his site to interpret the published protocol and to drive his display (probably making use of existing graphics programming facilities at his site). This might permit a convenient division of labor between the computer executing the application program and the computer driving the display (in pursuit of true resource sharing); it might be able to achieve very smooth performance despite poor network response, etc. The disadvantage of this approach is that the user must write a new program to interface his display to each different application protocol. In addition, there is no guarantee that the protocol required by the application program can actually be implemented within the user's operating system and display hardware.

Another approach is to develop one "general-purpose protocol" that tries to provide facilities that a large number of application programs could use and that a large number of user sites could interpret. Thus one user program can be used to interface to a number of application programs. The disadvantage of this approach is that the generality may preclude adequate response through the network, or that some application programs will find the general-purpose protocol too restrictive to be used at all. In addition, the design of such a protocol is not easy: attempting to provide a common device-independent framework for driving hardware of qualitatively different capabilities may be very difficult.

The protocol proposed in this document is, of course, a general-purpose protocol. However, we anticipate that special-purpose protocols will be absolutely necessary for certain applications. In these cases, the general-purpose protocol can perhaps be used as a starting point for development of special-purpose protocols. The general-purpose protocol should be used whenever possible, however, so that separate user programs need not be written for each user site.

The network protocol considered in this document is not intended to satisfy all graphics needs, for all terminals, now and in the future. It is limited to calligraphic pictures, to moderate interaction demands, and to "best effort" attempts to generate the required graphics. Certainly the impact of video technology will increase demand for variable character sets, shading, and maybe even animation techniques. These uses are clearly too new to consider in the present protocol.

### II.1. A Typical Session

At the beginning of an interactive session, a (human) user sits down at a graphics display attached to a computer on the network (the user host, UH; see Figure 1). He uses a keyboard associated with the display to communicate with the user host and to initiate a program that is dubbed the "user program," UP. This program initially performs TELNET\* functions, allowing the user to establish a connection with another host on the network where a service he desires is offered (the server host, SH). The TELNET functions can be used to log in, query system status, and so forth, and eventually to initiate a graphics application program (AP) in the server.

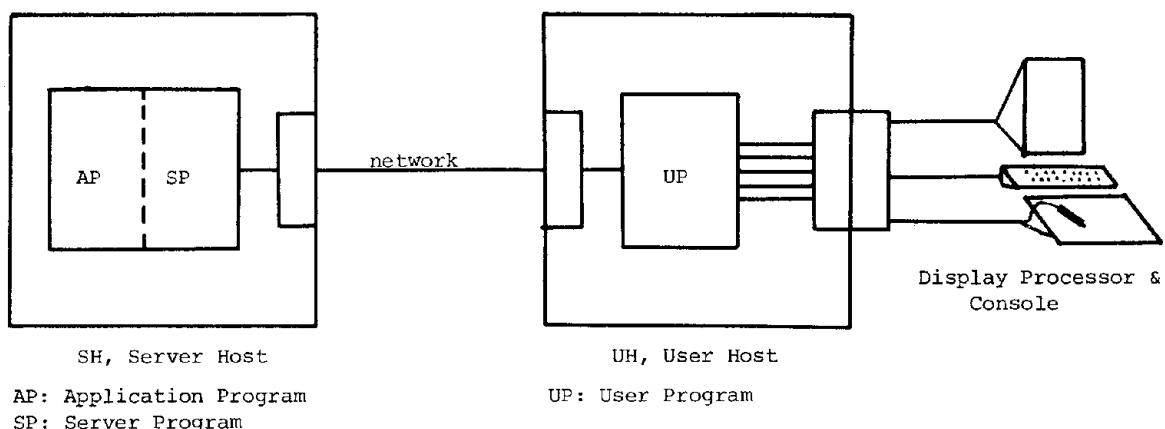


Figure 1: A network graphics session. A graphics application program running in one computer (the Server Host) drives a display console attached to another computer (the User Host).

The application program, when it wishes to produce graphical output or to request graphical input, makes use of a server program (SP) which may simply be a subroutine package. The job of the SP is to interface to the network graphics protocol on one side and to a "graphics language" on the other. (We shall use the term "graphics language" loosely here. It may just be a set of subroutine calls. The reason for making the distinction at all will appear below.)

The protocol transmitted between the SP and the UP is the graphics protocol described in this document. It provides facilities so that:

1. The SP can cause images to appear on the user's display screen.
2. The UP can report to the SP any interactions that the user initiates, such as keys depressed on the keyboard or stylus interactions.

---

\* TELNET is the name of a class of programs provided by many sites on the ARPA network that allow users to log in on time-sharing or multi-access systems at other sites in the network. The term TELNET also refers to the message-transmission protocol used by the network to accomplish this function. (See L.G. Roberts and B.D. Wessler, "Computer Network Development to Achieve Resource Sharing," AFIPS SJCC Proceedings, Vol. 36, May 1970, p. 543-597.)

3. The SP can discover various special properties of the user's display terminal, and can take advantage of them in conjunction with the application program.

We shall dub these types of protocol transmission "output," "input," and "inquiry."

The job of the UP is to interpret the protocol and to generate appropriate device-dependent information so that the image shown on the user's display corresponds to that specified by the SP using the protocol. It also implements the input and inquiry aspects of the protocol. The UP may have many "local options" that are not covered by the protocol. For example, the UP might contain facilities for creating hard copies of the image on the display without engaging in any protocol exchanges.

If the display terminal is attached to a TIP\*, the organization changes very little. The TIP is not the "user host," but only provides communications between the UH and the display terminal itself. In this case, the UH is often dubbed the "last intelligent host." Note that nothing prevents the UH and SH from being the very same computer.

## *II.2. A Model for the Network Graphics Protocol*

The analog of the task of defining a general-purpose graphics protocol is that of designing a "general-purpose interactive graphics system." This activity has been pursued by graphics system designers for years. Of course, these designs have been single-site designs; issues of device-independence and networking have been rarely addressed.

The philosophy behind the network graphics protocol can be demonstrated by giving a model of a general-purpose graphics system and combining it with the network model of Figure 1. The model of the system is shown in Figure 2. (The model is described in detail in [N&S] and [10GR]; we present only a brief description here.) To avoid misunderstandings, and for the sake of defining terminology, it is worthwhile to describe briefly each of the elements of Figure 2:

1. The input devices -- keyboard, stylus etc. -- are used by the operator of the application program to provide data and to control the program during execution.
2. The application data structure contains data, basically non-graphical, relating to the application program.
3. The input routines receive data from the input devices, make appropriate changes to the application data structure, and hand control to other routines.
4. The non-I/O routines analyze and modify the application data structure.

-----

\* TIP is the name of a particular kind of small computer attached to the ARPA network that implements TELNET functions for a number of dial-up communications lines. The TIP performs no significant computations for any users that dial it, but simply interfaces the network TELNET protocol to a number of terminals.

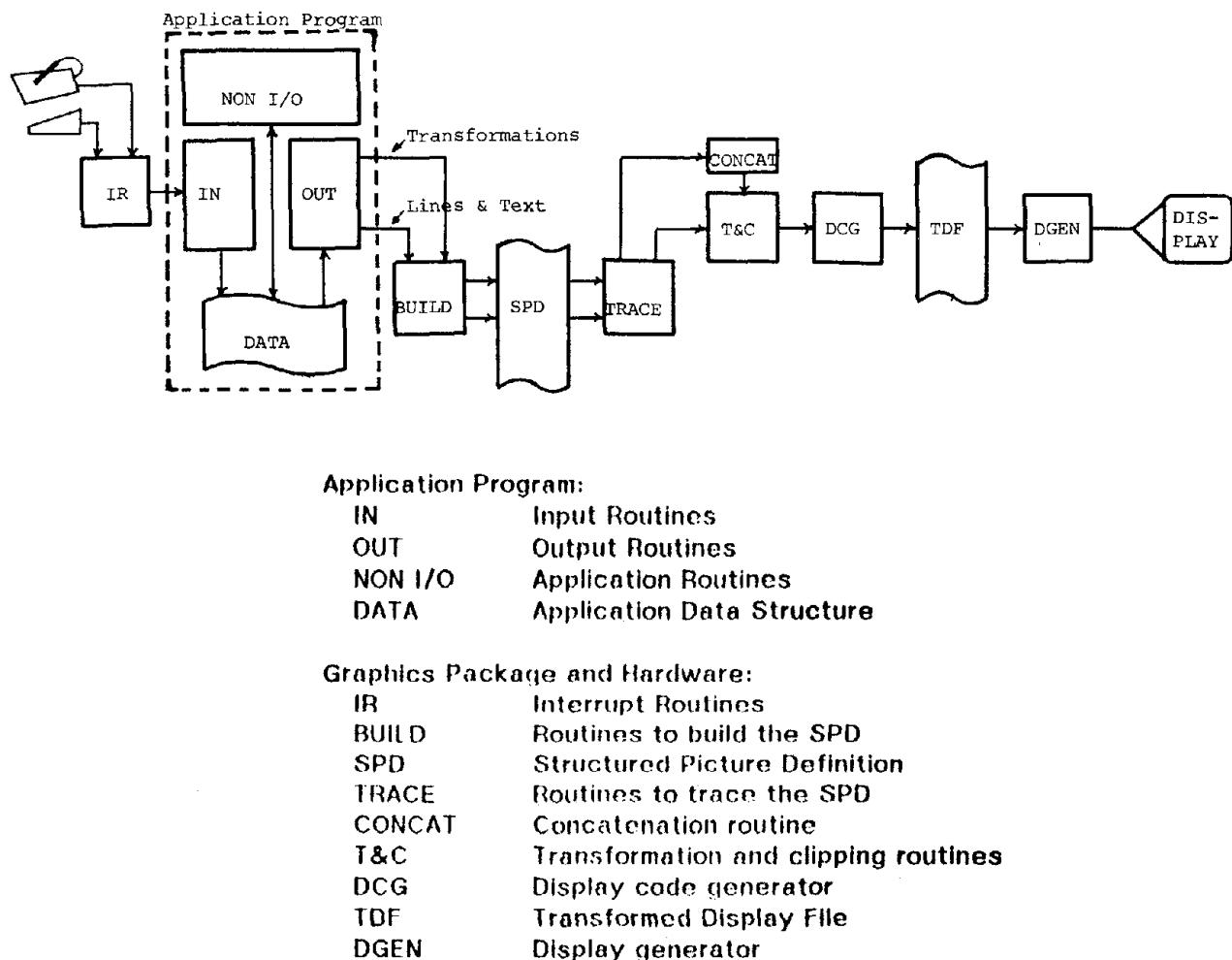


Figure 2: A Conceptual Model of a Graphics Application Program, a Graphics Package and Display Console.

6. The output routines build a structured picture definition from data drawn from the application data structure. Effectively they define how this data may be visualized for display purposes.

6. The structured picture definition defines the entire picture to be displayed, part or all of which may be visible on the screen. The picture definition is generally made up of a number of elements, representing parts of the picture known collectively as subpictures; subpictures may themselves be made up of other subpictures. A familiar type of subpicture is the symbol which is often used many times within a single picture or subpicture. Each reference or call to a subpicture element may denote a transformation -- scale, rotation, etc. -- to be applied to the subpicture.

7. The transformation routines apply the transformations specified in the structured picture definition and clip out information that lies

off-screen. Often an arbitrary rectangular viewport is used as the clipping boundary instead of the screen edge. These routines also handle the concatenation of transformations necessitated by multi-level structured picture definitions.

8. The transformed display file is essentially what remains of the picture after transformation and clipping. It is defined in the screen's coordinate system and is generally stored in a format that allows direct refresh or regeneration of the picture. The display file contains "primitives" that specify lines, dots and text to be displayed.

9. The display generator generally includes a vector generator and a character generator, which transform the contents of the transformed display file into signals that the display's deflection system can understand.

#### 10. The display itself.

We shall now analyze Figure 2 to see how the system can be implemented. The diagram illustrates three information structures (an application data structure, the structured picture definition, and the transformed display file), and three processes (the output routines, the transformations, and the display generator). The design of a general-purpose graphics system requires specifying the roles of all of these elements, with the exception of the application data structure. Our examination amounts to asking: what can the hardware do, and what does the graphics programmer think he can do? (For a more careful treatment of the concepts discussed in the next few paragraphs, see [10GR].)

Most display hardware is "processor" hardware, capable of implementing some or all of the three processes in Figure 2. If, for example, a display processor is available that implements transformations and display generation, then the transformed display file is absent, and we can view the hardware as "an interpreter of structured picture definitions." If the available hardware has fewer capabilities (e.g. no transformation ability), the picture is refreshed from the transformed display file, and the hardware is "an interpreter of transformed picture definitions." Or, if the hardware is a storage-tube terminal that does not require a display file for refreshing purposes, part of the display generation process is a software process. (However, a display file is still required, as we shall see below.)

Determination of what the programmer can do is somewhat more subtle, because the graphics system designer has complete control of the facilities he presents to the programmer. For example, the programmer of the system shown in Figure 2 would think he was creating a structured picture definition: the output routines allow him to create, modify, and delete elements of that structure. The remaining processes (transformation and display generation) and structures (transformed display file, if it exists) are inaccessible to the programmer; in some sense, he thinks that a "display processor" is interpreting the structured picture definition in order to generate a display. He cannot determine whether a transformed display file exists, nor whether transformations are done in hardware or software, etc.

If the structured picture definition is deleted, the programmer sees a quite different set of facilities. The output routines create (after transformation) a transformed display file. The programmer now thinks that the hardware is a "transformed display file interpreter." Again, the programmer is unaware of the

details of the display generator process (for example, if the display is a storage tube terminal, the display generator is a combination of a software display-file interpreter and some hardware vector and character generators. If, on the other hand, the transformed display file is used to refresh a display, the display generator is presumably entirely in hardware).

The discussion suggests that relative device independence can be provided if we permit two different kinds of output information: (1) information for building structured picture definitions, or (2) information for building transformed picture definitions directly. The first of these is matched to high-performance displays such as the LDS-2 or LDS-1; the second to standard refresh displays such as the IMLAC or GT-40.

How does this relate to network protocol? We can essentially view the UP as a "display processor," i.e. an "interpreter of structured picture definitions," or an "interpreter of transformed picture definitions." The network protocol is used to build and modify a picture definition contained in the user host. Various UP options are shown in Figure 3 (the dotted lines surround those processes that are implemented with display processor hardware). Figures 3a and 3b show the network used to transmit transformed information; the UP uses this information to build a display file for refreshing a display or for updating a storage-tube. Figures 3c, 3d and 3e show the network being used to transmit information for building a structured picture definition; the UP is an interpreter of a structured display file.

Figure 4 illustrates some possibilities for the server; the server can generate either structured or transformed display information. In Figures 4a and 4b the programmer "sees" a structured picture definition; in Figure 4c a transformed picture definition.

In the protocol, the UP tells the SP which kinds of format it can implement. It is perfectly possible for the UP to implement *both* -- the display images due to each of the formats are merged onto the screen.

The two different kinds of output format can be summarized as follows:

#### Transformed

The protocol is used to build and modify a set of "segments" (sometimes called "records") of a transformed display file, stored in the user host. A segment is a list of graphical primitives that specify lines, dots and text to be displayed at specific positions on the display screen. Individual segments may be deleted or replaced; they may be added to or removed from a list of segments to actually display on the screen (this is called "posting" and "unposting" segments). If a picture is composed of many segments, changes to the picture can often be made by replacing one or two segments; thus segmenting the display file helps to reduce the amount of information that must be transmitted through the network to effect a change. Considerable experience with this type of picture definition has demonstrated that device independence can easily be achieved [Omnigraph and Omnigraph.brief].

One advantage of transformed format is that the UP can be kept very simple; no transformations need be performed in the UH. A UP along the lines of Figure 3a should be able to be implemented in an IMLAC. The burden of transformation is left to the SH, presumably a large computer quite capable of being programmed to do transformations.

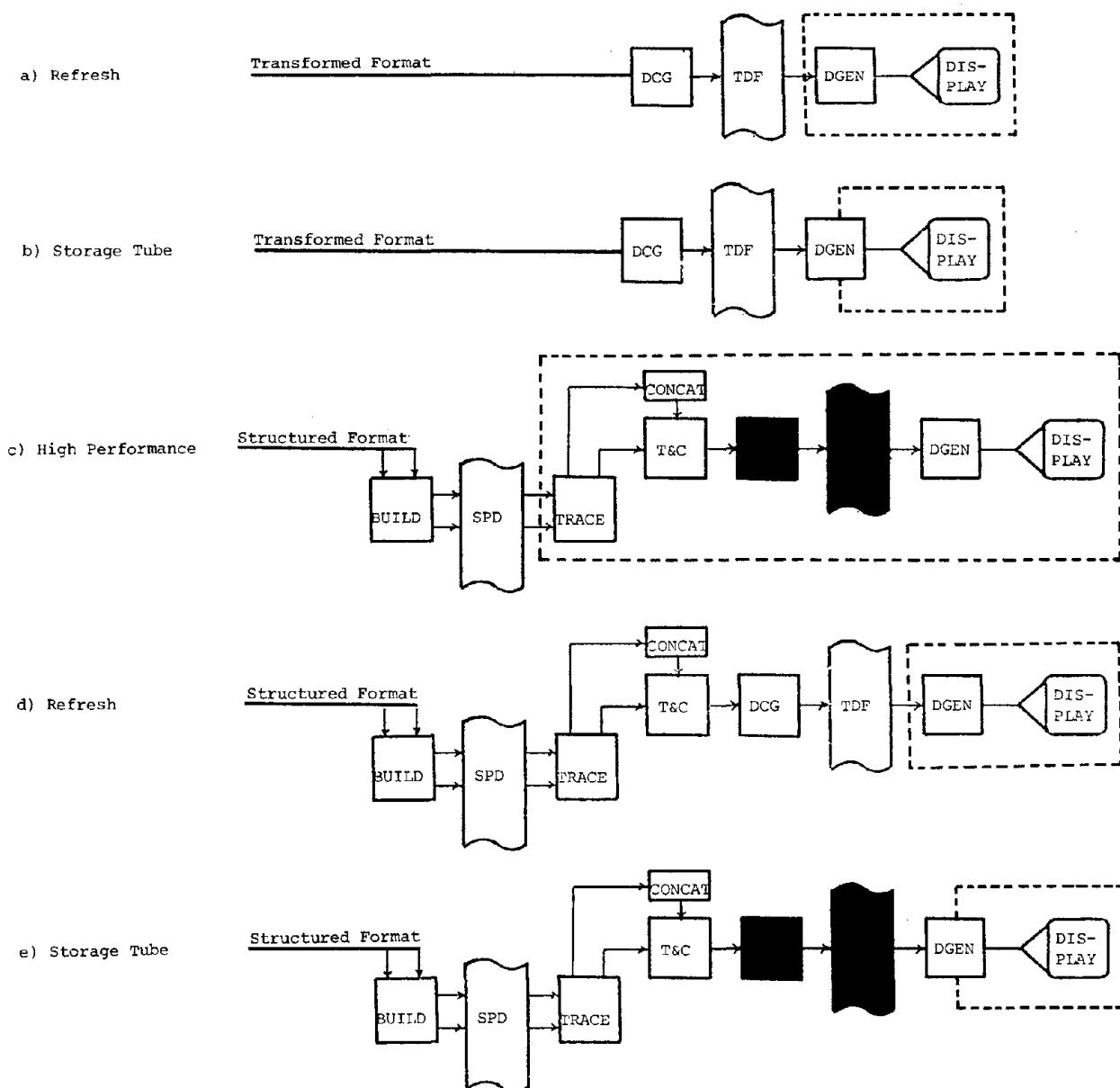


Figure 3: Various Configurations for the UP. Dashed lines surround processes implemented with hardware. Blackened processes are omitted in the particular configuration.

### Structured

The protocol is used to build and modify "figures;" each figure is a collection of "units." A unit may be a list of graphical primitives, such as lines, dots and text; or it may specify a "call" on another figure, together with a transformation to apply to the called figure. The protocol can replace individual units; altering a figure that is called in several places may cause widespread changes to the visible display. The structured format requires (in principle) even less network bandwidth for updates than does the transformed format; many updates may involve changing a

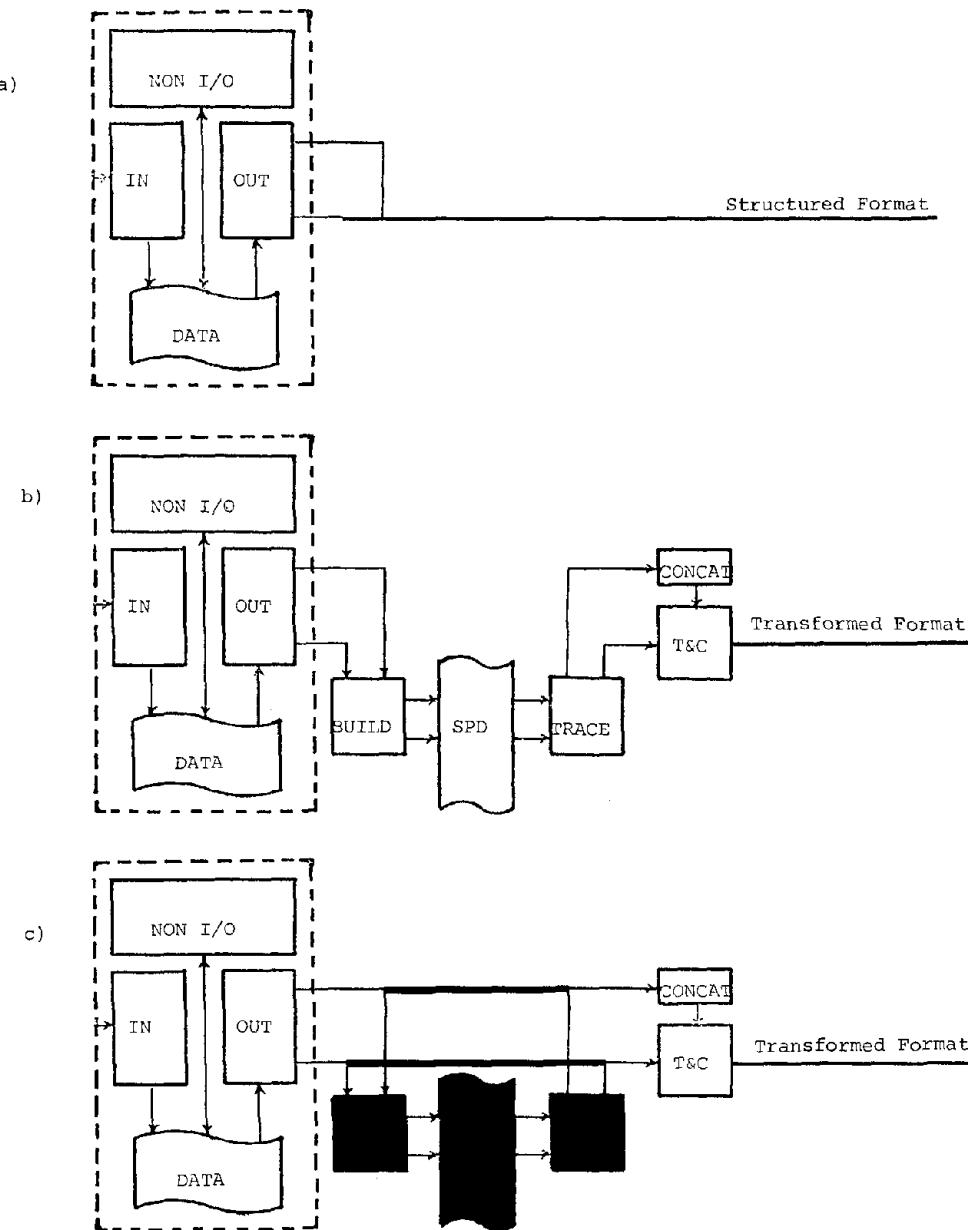


Figure 4: Various Configurations for the SP. These mate with the UP configurations of Figure 3.

single transformation (e.g. to change the viewing transformation for a three-dimensional object).

Although a structured picture definition can be interpreted directly by a few display processors that have transformation ability, most UP's that implement structured format will perform the transformation in software (Figure 3d,e; [10GR]). This implementation is relatively difficult, and certainly requires a fairly powerful UH computer.

The kind of network protocol (structured or transformed) should be clearly distinguished from the graphics language and the "output routines" of Figure 2. For example, the application data structure might be elaborately structured (e.g.

a circuit diagram, with instances of flip-flops and gates, etc.), and the programmer may think of the display generation in a structured way even though the network traffic is transformed (i.e. unstructured). This effect can be achieved with the display procedure technique.

Another way of looking at this is as follows: if we view the UP as a "display processor," then we might view the SP as a "graphics package" for driving the display processor. Most graphics packages attempt to insulate the programmer from the vagaries of a particular display processor, to provide structure even though the display processor does not, etc. In other words, the application programmer may have a structured view of the world even though the UP cannot provide such a view.

We have glossed over a significant problem introduced when some of the processes of Figure 2 are implemented in software. If the programmer thinks he is producing a structured picture definition, but the hardware is interpreting a transformed picture definition, when is the transformation software run? A similar problem occurs in Figure 3b: when is the display-generation software run in order to update the display? One answer is: whenever a display file is changed, necessary software processes are invoked to update the display. This has several bad effects. A more useful technique is for the SP (and application program) to signal the UP whenever a collection (or batch) of changes is complete and the display should be updated. This technique has the following advantages:

1. Transformation software (e.g. 3d, 3e) is executed only when necessary in order to update the display; we never needlessly repeat transformations.
2. Screen erasures on storage tubes (e.g. 3b) can be minimized -- we need erase the screen a maximum of once per batch of updates, rather than once per update. See [Omnigraph.brief].
3. All changes to the display appear "instantly." Network speed means that display-file updates will arrive at the UP over a longish period of time. If the effect of these changes is delayed until a batch is finished, the display will appear to change "all at once," a much more satisfying effect than many slow changes. This is particularly true if an image is being replaced by another image that is a scaled-up version of the original: some things grow before others, and the display passes through a number of nonsensical states.

To take full advantage of this technique, the AP should specify when the screen should be updated to represent precisely what is specified in the display file. These "end batch of updates" commands should probably precede each request for new user input -- thus the user will see an up-to-date image before formulating his response. Specifying this information is quite easily done, and is not at all unnatural for the application programmer.

If this policy of delaying changes is not to a programmer's liking, the SP could be instructed by the AP to issue an "end batch of updates" command following each and every update to a segment. Updates only occur as a result of a small number of protocol commands (see section III); this should not be difficult.

### *II.3. Positioned Text*

The graphics facilities already described can be used to put text information on the screen. However, certain application programs display exclusively text (e.g. NLS [NLS]) and require a rather different set of facilities for controlling the display. Strings of text are "positioned" on a display screen and "edited" by commands from the server host.

The positioned text facilities have been separated from the graphics facilities for two reasons: (1) users with simple alphanumeric terminals (e.g. Hazeltine) can in fact implement the positioned text protocol even though they cannot implement the full graphics protocol; (2) it simplifies the design of user programs that only need to provide text interfaces (e.g. allows one to build a UP in an IMLAC that is optimized for NLS use).

This graphical output format is completely independent of the transformed and structured formats. A UP may report to the SP that it implements only the positioned text format -- this might be the case of a UP for NLS use.

### *II.4. Input Facilities*

The problem of providing input facilities is even harder than that of providing output facilities. The difficulties are chiefly those of device independence and of adequate performance. The device independence issue is easily demonstrated: display hardware can have a large number of very different kinds of input gadgets attached (light pens, tablet and styli, joysticks, knobs, buttons, etc.) that have different properties and different methods of reporting their output (e.g. periodically, on computer demand, or "when something changes"). In addition, operating systems at user sites often enforce restrictions on the use of input equipment in order to avoid undue system degradation.

The problem of performance is nicely demonstrated by Figure 1 -- If each input must be shipped to the server host, processed by the AP and SP, then any display updates shipped to the user host and processed by the UP before the user sees the response, it would be impossible to use many interactive graphical techniques.

The protocol attacks these problems in simple and probably inadequate ways. For this reason, the input facilities are the most controversial and experimental of the protocol.

The device independence issue is solved by Inquiry: the UP reports to the SP a list of available devices. The SP and AP can then collaboratively arrive at an acceptable set needed for operating the AP. If the set of input devices is insufficient, the AP can perhaps engage in a dialog with the (human) user to seek remedies. Perhaps a spare device can be plugged in; perhaps another version of the UP can be run which implements the required device. Or, if the AP and SP are sufficiently flexible, perhaps the command language of the application program can be altered dynamically to permit its operation with the available devices. For example, if the UP responds that it has no coordinate input device (and that it is not willing to simulate one with, say, two knobs) then the AP might want to use a keyboard-based interaction sequence and to organize the entire command system differently.

The performance difficulties are addressed by permitting the SP to ask the UP to

use a particular interactive technique in conjunction with an input device, and to report the results of the interaction. We shall term such **interaction techniques events**. The techniques often involve providing "local feedback," so that the user sees the results of his interaction without a long network delay. Examples are: displaying a "tracking dot" at the current location of a coordinate input device, or displaying a trail of "ink" behind the tracking dot, etc. Examples of the special events that the protocol provides are:

**Positioning** -- Using a coordinate (or other) device to provide a pair of coordinates to specify the position for something. The UP will usually display a tracking dot to aid the user in coordinating his input with the display.

**Pointing** -- Using a coordinate (or other) device to identify an object currently being displayed on the screen. Again, the UP will probably provide tracking. There are two ways of providing this technique: one is to assume that the display terminal has some hardware feature that aids identifying a graphical feature being pointed at (e.g. light pen or comparator). However, the same information can be deduced from a positioning interaction and some software calculation (either in the SP or UP) to determine what object is being identified. Thus, even if a UP cannot provide the "pointing" interaction, the SP may be able to (see [N&S] for a description of the process).

**Stroke** -- Using a coordinate (or other) device to trace out a free-form curve and reporting a stream of coordinate points on the curve. The UP will provide tracking and leave a trail of dots ("ink") along the curve.

**Dragging** -- Using a coordinate (or other) device to cause some portion of the display image to move in synchronism with the coordinate device. This technique could be classed as "highly interactive," and some display terminals cannot provide it.

The protocol provides the SP with two basic methods for dealing with input devices: (1) to request and obtain the state of an input device, e.g. the current position of a coordinate input device, and (2) to enable various events, and to obtain a "report" describing the events resulting from user actions.

The user site has a wide latitude in implementing these input facilities. Since inquiry is used to find out what devices and events the user site implements, the site may implement as many or as few as it likes. The latitude permits individual users to use devices differently or to establish special feedback mechanisms (e.g. a number displayed on the screen that represents the current reading of a knob). It also permits user sites with weird hardware or operating systems to emulate input devices in any way they choose. (For example, no requirements are put on sampling rates for inked strokes; something "reasonable and proper" is adequate.)

#### II.5. Inquiry

The protocol has no set of "standard" features; there is thus no standard graphics terminal as viewed by the protocol. The inquiry function is used to transmit to the SP a certain amount of detailed information about the terminal in use and the UP that drives it. This information is a "constant" that will probably be requested by the SP when it initiates a graphics session.

The information transmitted by the inquiry response is in part for information only. However, some of the information returned is essential in order for the SP to transmit legal protocol to the UP. In outline, the information returned is:

List of implemented protocol commands. This list tells the SP whether the UP implements transformed format, or structured format, or positioned text, or any combination of them, and so forth. In addition, this report tells which optional parts of the protocol are implemented by the UP.

Coordinate Information. This information is necessary for the SP to carry out transformations that generate coordinates in the coordinate system used by the terminal.

Parameters that describe available character sizes, available intensity resolution, available line textures, etc.

A list of available input devices and events. A "device number" is specified for each device; this is used when reading the state of a device. Similarly, an "event number" is specified for each event, and is cited when enabling or disabling it.

An ASCII text string that describes the terminal, e.g. "IMLAC PDS-1 in room 22."

The information in the inquiry response (that transmitted from UP to SP) that is not essential to further protocol operation may still be useful to the SP in order to drive the terminal intelligently. For example, inquiry can determine the kind of display being used, not so as to send device-specific code to it, but so that the AP does not try to use a graphic technique on a terminal that cannot handle it (e.g. some sort of dynamics on a storage tube).

## II.6. UP Implementations

Although the description of the protocol is quite lengthy, the protocol itself is quite simple. The aim of the design is that the UP could be implemented in an IMLAC or GT-40 or similar "smart" terminal. Of course, it could also be implemented on any host computer, with a less smart terminal attached to the host.

There are three main mechanisms for simplifying the implementation: (1) the inquiry function specifies many of the terminal details to the SP, thus freeing the UP from coping with complicated logic to implement complicated operations; (2) much of the protocol is optional, a subset being quite adequate for most applications; (3) in thorny areas (e.g., input protocol), the protocol is deliberately vague, allowing the UP implementation considerable latitude to obey the protocol as best it can.

The philosophy of "making the SP drive the terminal," rather than making the UP achieve an ideal performance is key. This approach puts ingenious graphics programming and command languages where they belong, in the server.

## II.7. Summary

Here is a brief summary of the main philosophical points of the protocol:

- O. The protocol is based on the premise that much, but not all, graphics can be done within a "general-purpose" framework. Special-purpose protocols are inevitable.
1. The protocol facilities for graphical output can be likened to those of a graphics system driving a display processor. The protocol creates and modifies a display file at the user site.
2. The protocol provides options: the SP and UP must agree on what kind of "display processor" the UP can implement (Structured, Transformed, Positioned Text, or some combination) and on selection of input devices. The protocol thus implements no fixed "virtual display" but rather a large variety of display processors.
3. Portions of the protocol are left deliberately vague. The user program is expected to implement features to the best of its ability; if the implementation is inadequate, the shortcomings will probably be quickly discovered by a user.
4. Although the protocol appears largely "device independent," inquiry functions permit the application program to discover many hardware details necessary to drive the display intelligently.

### SECTION III

#### *The Protocol*

This section presents details of the graphics protocol. The topics covered are the connection protocol, the graphical output protocols, the graphical input protocol and the inquiry protocol.

The section describes network traffic as a series of commands and operands, all expressed in a common notation. The smallest unit of traffic is an 8-bit byte. The construct <...> refers to a specific byte, i.e. a command that is assigned a particular op-code (listed in the appendix). Constructs of the form <\*...\*> refer to sequences of bytes that are defined elsewhere in this document.

Following are some standard definitions:

All numbers in this document are decimal unless preceded by an apostrophe, in which case they are octal (8='10).

A <*\*small.integer*> is one 8-bit byte that contains the integer (range 0 to 255).

A <*\*large.integer*> is two 8-bit bytes that together comprise a 16-bit integer. The first byte transmitted is the high-order 8 bits; the second the low-order 8 bits (range 0 to '177777).

A <*\*small.fraction*> is a two's complement fraction in the range [-1:1-1/128]. It is defined as (<*\*small.integer*>-128)/128.

A <*\*large.fraction*> is a two's complement fraction in the range [-1:1-1/32768]. It is defined as (<*\*large.integer*>-32768)/32768.

A <*\*count*> is either one or two 8-bit bytes, depending on the size of the count. If the count is less than or equal to 127, then <*\*count*> is simply one byte that contains the count; otherwise it is two bytes, and the count is computed as (byte1-128)\*256+byte2.

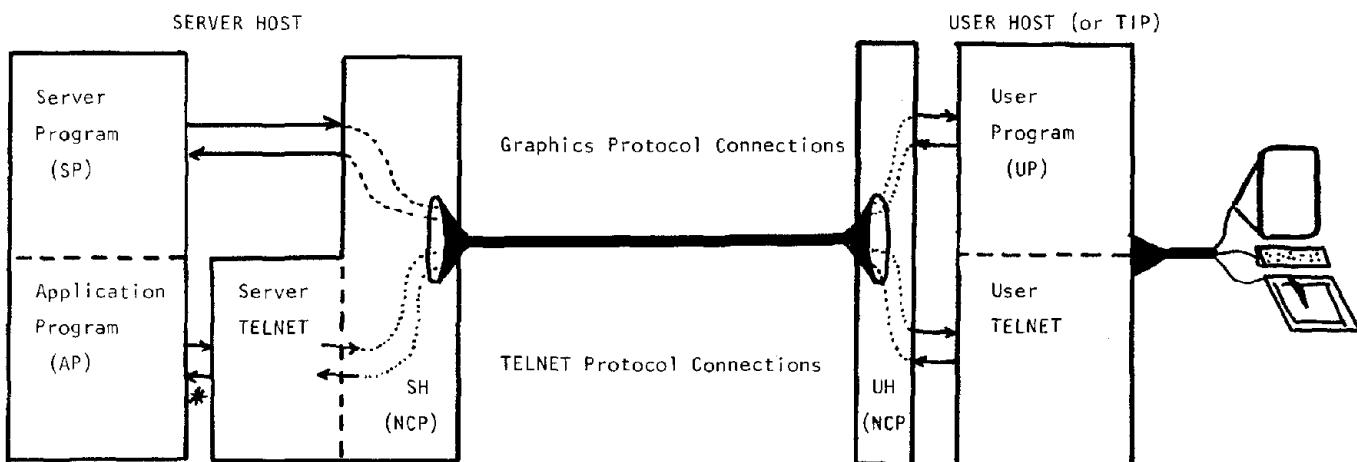
A text string <*\*text*> is defined as a character count, <*\*count*>, followed by that number of 8-bit bytes. If the text string is intended to be interpreted as ASCII text, then the following conventions are observed: (1) every printing character in the ASCII set is in the set (high-order bit is zero), (2) the ASCII control characters carriage return ('15), line feed ('12), tab ('11) and formfeed ('14) may appear; (3) codes in the range '200 to '377 may be used for whatever purposes user and server desire, but the protocol establishes no conventional meanings.

The various forms of picture definitions in the user host are given names (e.g. <*\*seg.name*>, <*\*ptext.name*>). These are all defined as 16-bit quantities, in <*\*large.integer*> format. The name spaces are all separate.

### III.1. Initial Connection Protocol

This section describes the mechanism for establishing connections between the SP and UP in the ARPA network. An understanding of this section is not required in order to understand the remainder of the protocol.

A user session with a graphics application program (see Figure 5, an elaboration of Figure 1) has many close analogies to a TELNET session with a program: the user must log into a server system, execute several system commands, initiate the program, communicate with the program as it is running, perhaps interrupting a running program, logging out, etc. A graphics session certainly requires all of these features; in addition it will require a pathway for transmitting graphics protocol information in both directions.



In the case of an intelligent terminal attached to a TIP, the boxes labeled UH(NCP) and User TELNET are implemented by the TIP; the box labeled UP would be implemented in the intelligent terminal.

\* This communication path is accomplished with operating-system calls that type characters on the controlling terminal and that accept characters from that terminal. (That is, this is not a direct network connection.)

Figure 5: A Network Graphics Connection and Associated Processes

The graphics protocol traffic is inherently separate from the TELNET traffic, and should use a separate network connection. It is inconvenient to multiplex graphics protocol on the TELNET connections because many operating systems give AP's access to the TELNET connection only via the mechanisms that they use to access a controlling terminal: the application program "types" to the

terminal. In addition, the TELNET connection is used by some operating systems when signaling errors; this summary use might interfere with a multiplexing scheme. Thus, the protocol requires a pair of network connections dedicated to graphics traffic.

This approach can also accomodate intelligent terminals (e.g. IMLAC's) attached to TIP's. If the IMLAC program interprets (1) ASCII text and (2) graphics protocol, and uses a simple multiplexing scheme to transmit these two kinds of traffic to and from the TIP, then the TIP can support the traffic with minor modification. The TIP need only be able to establish the extra pair of connections and to multiplex the two kinds of traffic for consumption by the intelligent terminal. (This is *not* the same as multiplexing the TELNET connection.)

There are two connection schemes, one to be used for now, and one that is intended for use when the new style TELNET protocol is fully operational and when a sufficient number of operating systems have been modified to give the UP and SP access to the negotiation mechanism.

For now.

When a graphics SP is started (usually by the user's issuing an appropriate command to the user host through the TELNET connection), and wishes to initiate a graphics dialog, it gets a pair of complementary socket numbers from its operating system. These are called SP-send and SP-receive. The SP then "types" over its TELNET connection an ASCII string that consists of 17 characters: the first six characters are \*GICP\*; the remaining 11 characters are the 11 octal digits (in ASCII, of course) of the socket number for SP-receive. Note that SP-receive is an even number, and that SP-send = (SP-receive)+1. (This is a network convention that seems quite nice for now.) After typing this information, the SP does "listens" on those socket numbers.

The UP recognizes the special character string and following digits, and issues RFC's (requests for connection) from two of its sockets (UP-send and UP-receive) to the complementary sockets at the SP. The SP will return the RFC's, thus completing the connections. The dust has settled, and the connections UP-send=>SP-receive and SP-send=>UP-receive are ready for use.

Ultimately,

When reason descends on the world, the TELNET negotiation mechanism (see NIC 15372) will be used to establish the willingness to transmit graphics information (DO, DONT, WILL, WONT GRAPHICS), and a subnegotiation transmission will transmit the socket number. In the case of an intelligent graphics terminal attached to a TIP, the TIP TELNET responds to the negotiation and establishes the connections.

Even after graphics protocol has been initiated, not all communications between SP and UP will be graphics protocol; there still may be TELNET traffic. From UP to SP will go "breaks;" from SP to UP will go text "typed" by the application program (rather than enclosed in some graphical command for displaying text) as well as system error or informational messages. Such SP-to-UP text is called "unescorted text." The user will probably wish to read this text, since it is often important for understanding or operating the AP. The text can be handled by the UP in either or both of:

1. Show it on the display screen. This option is controlled by the graphics protocol (see positioned text).
2. Local (UP) option. This treatment is up to the UP; the text can be displayed regardless of provisions of method 1; it can be typed on an adjacent hard-copy terminal, or whatever.

### III.2. *Output Protocol Formats*

The network graphics protocol makes provision for three kinds of formats for graphical output:

- 1: Transformed format.
- 2: Structured format.
- 3: Positioned text format.

It is possible to design a UP that can correctly interpret any combination of formats coming from the SP. The UP reports to the SP, via the inquiry response, which formats are implemented (this information is contained in the list of implemented commands).

The design of the structured format is still in preliminary stages. This is chiefly because of difficulties designing a suitably device-independent view of transformations (e.g., clipping and rotation conflicts, providing for the constraints of transformations performed with analog hardware). A brief description of the preliminary design appears in the appendix.

### III.3. *Transformed Format*

The protocol for "transformed format" is used to build and modify a set of segments of the picture definition stored in the UH. All coordinate transformations are performed in the SP prior to sending data to the UP; the segmented picture definition thus contains descriptions of lines, dots and text that will have a fixed location on the screen.

A segment is created in the following fashion. The "open segment" command is sent to the UP, together with a "name" for the segment. Any subsequent graphical primitives (e.g. line, dot, text) sent to the UP are added, in order received, to the currently open segment. The creation process is terminated by a "close segment" command. The segment now specifies how to draw some (or all) of the desired display image.

The creation of a segment simply specifies a list of graphical primitives and not the use to which they are put. If the segment is to be displayed, a "post" command specifies that a segment is to be added to a list of segments to be displayed. The "unpost" command removes a segment from the display list. The "kill" command is used to destroy the segment altogether.

No changes to the visible display are made until the "end batch of updates" command is received at the UP. Thus, the effects of the "post," "unpost," and "kill" commands must be delayed until this command is received.

Although the graphical primitives that are contained in a segment cannot be

modified, a certain number of "attributes" associated with each segment may be individually modified. These facilities are described more fully below.

#### *III.4. Segment Control*

A detailed description of the commands follows:

##### **<seg.open> <\*seg.name\*>**

This command opens a new segment and specifies its name. All subsequent graphical primitives will be added, in order received, to the open segment. Graphical primitives need not follow contiguously -- other graphics protocol commands may intervene between specification of primitives to be added to the segment. If a segment with the same name already exists, it is not destroyed at this point (this technique is called "superceding" a segment; the protocol insists that the segment being created is double-buffered).

Immediately after the **<seg.open> <\*seg.name\*>**, any attributes that are to be associated with the new segment must be set, before the first graphical primitive is specified. This operation indicates to the UP which attributes of this segment might be changed later on. Such attribute settings are accomplished with the **<\*attribute\*>** sequence, described in section III.10. The reason for this convention is that the UP may wish to build the display file in a slightly different way if certain attributes are specified, so that they may later be changed.

##### **<seg.close>**

This command signals the end of generation of (or appending to) the currently open segment. The segment can now be posted, unposted, or killed.

##### **<seg.post> <\*seg.name\*>**

The specified segment is added to the list of segments to be displayed. If the named segment is the currently open segment, it is "closed" first. No change is made to the currently visible display.

##### **<seg.unpost> <\*seg.name\*>**

The specified segment is removed from the display list. Again, no change is made to the currently visible display.

##### **<seg.kill> <\*seg.name\*>**

The specified segment is deleted entirely. If the segment is currently in the display list, it is "unposted" first. Note that this means deletion may be delayed so as not to alter the currently visible display until the "end batch of updates" command arrives.

##### **<seg.append> <\*seg.name\*>**

This command specifies that all subsequent graphical primitives are to be added to the end of the segment named, which must already exist. Note,

however, that even if the segment named is currently being displayed, the appended information cannot be displayed until the next "end batch of updates" command. A segment that is appended to leaves attribute settings unchanged. In particular, the set of available attributes cannot be augmented beyond those specified originally following the <seg.open> command.

The "append" feature is entirely optional; if the UP implementation does not permit appending, the inquiry response will so specify. Failing to implement this command has no effect on the interpretation of the rest of the commands.

#### <end.batch.of.updates>

This command specifies that a collection of updates is complete, and the visible display should be updated to reflect the changes. In particular:

- Any killed segments are entirely deleted and returned to free storage.
- The old versions of any superceded segments are deleted and replaced by the new versions.
- Any "posts" or "unposts" transmitted since the last "end batch of updates" can be performed.
- Any "appends" specified are actually added to the appropriate segment.

(If the display file is not used to refresh a display, as is the case with a storage tube terminal, the modifications to the display file structure itself need not be delayed until the <end.batch.of.updates> command arrives, but all screen changes must be delayed. This minimizes the number of full-screen erasures required on storage tubes.)

Some application programs may wish to cause changes to appear as soon as the change has been successfully transmitted to the UH (e.g. when a <seg.post> is sent). In this case, the AP or SP can simply arrange to follow each such modification command with a <end.batch.of.updates> command.

One drawback of delaying changes is that up to twice the amount of display-file storage can be consumed, compared to that required to store one picture. This will happen if a batch of changes involves superseding every segment. This might cause the UP to exhaust the free storage available to it for display files. If this happens, the UP may simulate the effect of an <end.batch.of.updates> command prematurely, and thus reclaim storage used for superseded segments. This should really be considered an error.

A number of anomalous or "illegal" sequences of the segment-controlling commands might occur. Specific remedies are described below. (A UP implementation is not required to follow these conventions, and SP implementations should not count on them.)

1. If graphical primitives are received by the UP when no segment has been opened (either by <seg.open> or <seg.append>) they are discarded. The UP might issue an error indication.

2. If a <seg.open> or <seg.append> is received when a segment is already open, the newly-received command is ignored. Again, the UP might issue an error.
3. If the segment named by a <seg.append> does not exist, the command should be treated as a <seg.open>.
4. If the segment named in a <seg.kill>, <seg.post> or <seg.unpost> does not exist, the command is ignored.
5. <end.batch.of.updates> may occur anywhere, even while a segment is open. Primitives added to the currently-open segment should not, however, be displayed (because the segment is being created, and is not yet finished!).
6. If the <seg.kill> or <seg.unpost> commands give a name that matches that of the currently open segment, the command refers to the old version of the segment (if any), not to the one currently open.

### III.5. Graphical Primitives

The following commands cause primitives to be added to the currently open segment:

```
<seg.dot> <*x.s.coord*> <*y.s.coord*>
<seg.move> <*x.s.coord*> <*y.s.coord*>
<seg.draw> <*x.s.coord*> <*y.s.coord*>
<seg.text> <text*>
```

These primitives are the familiar commands for adding points, lines and text to the open segment. No relative mode is provided: the SP can easily let the application program specify relative information, and convert to absolute for transmission.

### III.6. Coordinate Systems

The coordinate system used for arguments to dot, move and draw in the transformed format is called the "screen coordinate system." *The format of a <\* .s.coord\*> construct is determined from the inquiry response, and is in the coordinate system actually used by the graphics terminal.* The inquiry response defines how many 8-bit bytes are used to specify such a coordinate, and what values correspond to the left, right, bottom and top addressable points on the screen. (For a discussion of other possibilities for the screen coordinate system, see [NIC 19933].) See section III.16, item 2 for an example of the screen coordinate calculations.

### III.7. Intensity

The SP can select an intensity that is to be used for all subsequent graphical primitives added to segments (except as noted below). The command

`<set.intensity> <*count*>`

specifies the intensity in `<*count*>` format. Permissible values of the `<*count*>` are returned in the inquiry response. Exception: If intensity is specified as an attribute of a segment, that value overrides any that is specified within the segment. (See the section on attributes, below. One of the reasons for declaring the intention to change attributes when opening a segment is so that the UP can generate the segment in such a way that the attribute may be implemented efficiently on the display hardware.) A default intensity (anything visible) is established by the UP initially. \*

### III.8. Line Type

The SP can govern the type of line added to segments by the draw primitive. The sequence

`<set.type> <*count*>`

sets the line type for all subsequent lines. The Inquiry function can be used to find out how many distinct types (e.g., dotted, dashed) are supported by the UP. Type 0 is always a normal solid vector, and is established as the default type by the UP initially. The protocol makes no precise definition of line types.

### III.9. Character Display

The text primitive adds alphanumeric information to the open segment; most terminals will have hardware character generators for actually drawing the characters. However, the SP can control certain aspects of character display: size and orientation. There are two methods of specifying the size or orientation: (1) the SP may select one of several discrete character sizes and orientations available (details about these sizes are contained in the inquiry response), or (2) may select an exact size. All subsequent text primitives (up to the next time the SP asks to change) use that style.

Initially, the UP sets a default size (any legible size) and orientation (horizontal). The UP implementation should try to be resilient when the SP generates text that lies off the screen.

The orientation is selected by the sequence:

`<set.character.orientation.discrete> <*count*>`

where `<*count*>` specifies one of several discrete orientations available, as specified in the inquiry response. Alternatively, the following (optional) command may be used:

-----

\* All that is needed to implement this facility in the UP is one variable that maintains the "current intensity." Whenever a line, dot or text string is added to an open segment, the value of the variable specifies the intensity. A similar method is applicable for line type, character size, and character orientation.

`<set.character.orientation.continuous> <*large.fraction*>`

where the fraction is a measure of the angle between the text drawing direction and the horizontal as fractions of  $2\pi$ . The fraction for horizontal direction is 0, for vertical text running up is  $1/4$ , etc. If this command is implemented, the UP agrees to draw characters at any orientation.

The size is selected by one of two methods. A specific discrete size, the details of which are returned in the inquiry response, is set with

`<set.character.size.discrete> <*count*>`

where the count is the index of the character size returned by the inquiry response.

The following command can be used to specify a "continuous" character size. This command is optional; if it is implemented, the UP agrees to display characters of any size:

`<set.character.size.continuous> <*char.size.description*>`

where `<*char.size.description*>` is

`<*BW.s.coord*><*BH.s.coord*>  
<*CW.s.coord*><*CH.s.coord*><*CT.s.coord*>`

The coordinates specify five dimensions, in screen coordinates, of the character, as shown in Figure 6.

### III.10. Segment Attributes

Although the graphical primitives that make up a segment may not be modified, a small number of "attributes" may be. This section describes the attributes and the mechanisms for changing them.

Just after the `<seg.open>` command is transmitted, any number of `<*attribute*>`s may be specified; these are attributes that are to be associated with the new segment. At some later time, the `<change.attribute>` command can be used to change any attribute that was specified in this original list. The reason for the initial list is that the UP may wish to generate the display file for a segment differently if it knows that certain attributes might later be changed.

The attributes are individually optional. If the the UP does not implement the `<set.xxxx.attribute>` command, then it has no facilities for dealing with the corresponding attribute.

Attributes may be changed by the sequence:

`<change.attribute> <*seg.name*> <*attribute*>`

Such changes do not take effect immediately; an `<end.batch.of.updates>` command causes changes.

The possible `<*attribute*>` sequences are:

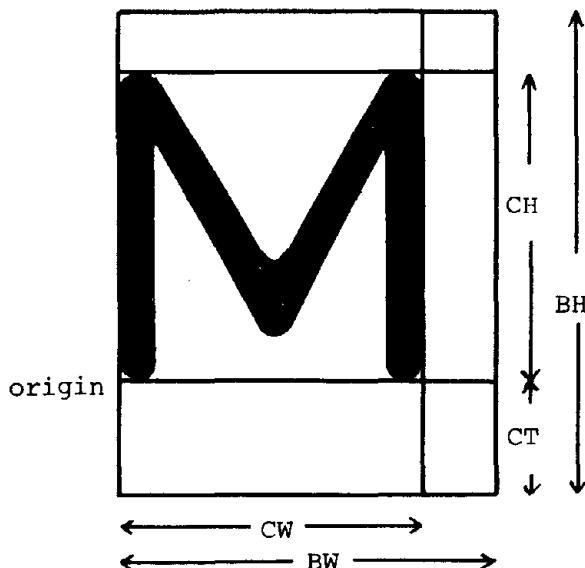


Figure 6: A Character Size Description. All measurements are made in the screen coordinate system. The point labeled "origin" is the reference point for the character, i.e. It coincides with the  $(x,y)$  point set by the `<seg.move>`, `<seg.dot>` or `<seg.draw>` previous to the `<seg.text>`.

**Highlighting.** This is an on/off condition for highlighting (e.g. blinking or intensifying unnaturally) an entire segment. The **relevant `<*attribute*>`** sequence is either

`<set.highlight.attribute> <on>`      or  
`<set.highlight.attribute> <off>`

The default is `<off>`.

**Hit sensitivity.** This on/off attribute is used in conjunction with Input facilities (see III.13). If a segment is hit sensitive, then its name may be reported to the SP if a coordinate input device is pointed at any line, dot or text that is part of the segment. The **`<*attribute*>`** sequence is either

`<set.hit.sensitivity.attribute> <on>`      or  
`<set.hit.sensitivity.attribute> <off>`

Default is `<off>`.

**Intensity.** This attribute controls the intensity of all graphical primitives in the entire segment. It overrides any `<set.intensity>` settings within the segment.

`<set.intensity.attribute> <*count*>`

The permissible values of `<*count*>` are returned in the inquiry response. Default is something visible.

The intensity attribute and `<set.intensity>` are mutually exclusive within a segment: a segment created with an intensity attribute will ignore any `<set.intensity>` commands; one created without an intensity attribute will honor all `<set.intensity>` commands.

**Screen select.** If a user site has several display screens, the SP can control which screens a particular segment is to be viewed on (see inquiry section for a description of how the SP can discover the number of

screens in use at the user site). Screens are enabled by a mask byte, with a bit for each of up to eight display screens (the high-order bit of the byte is the first display, the next the second, etc.). If the bit is on, 'posting' the segment will cause the segment to appear on the corresponding screen. The default is '200 (screen number 1 on).

`<set.screen.select.attribute> <*small.integer*>`

Initial position. This command sets the position on the screen of the first item in the segment. This indicates the intention of the SP to either cause dragging to be performed, or to change the position of the segment at some future time.

`<set.position.attribute> <x.s.coord*> <y.s.coord*>`

As an example: `<seg.open> <seg.name=2> <set.position.attribute> <coordinate=500> <coordinate=500> <seg.move> <coordinate=400> <coordinate=500> <seg.draw> <coordinate=600> <coordinate=500> <seg.post> <seg.name=2> <end.batch.of.updates>` causes a segment with one visible line to be created. The line extends 100 units left of the initial position, and 100 units to the right. If we later wish to change the position, the command `<change.attribute> <seg.name=2> <set.position.attribute> <coordinate=200> <coordinate=500>` will move the line 300 units to the left. Note that some UP's will wish to store segments that are to be repositioned in an internal format that is different from that for other segments (e.g., as a sequence of relative moves and vectors).

### III.11. Segment Readback

This section describes facilities for transmitting information about a segment stored in the UP back to the SP. This allows the SP to save "copies" of segments to use later or to drive hard-copy equipment. The facilities can also be used for debugging or for certain kinds of "group sessions" in which users at several points in the network may be working concurrently and wish to "transmit" pictures to each other.

This collection of facilities is optional. The implementation should, however, be very simple, and we expect that most UP's will offer this service.

The command sent from the SP to the UP

`<seg.readback.seglist>`

causes the UP to transmit to the SP the sequence:

`<seg.readback.seglist> <*count*> <*seq.name*> ...  
<*count*> <*seq.name*> ...`

The UP returns the count of posted segments, followed by their segment names, followed by the count of unposted segments, followed by their segment names.

The command from SP to UP

`<seg.readback.seg> <*seg.name*>`

causes the UP to send back to the SP a sequence in network format that completely describes the named segment. The sequence can be any legal sequence of segment commands described in this section (including character size and type control commands) sufficient to unambiguously describe the segment (i.e., if the sequence were transmitted to the same UP, it would generate an identical display). The sequence is <seg.open> <\*seg.name\*> followed by any attributes of the segment, such as <set.hit.sensitivity.attribute> <on> or <set.intensity.attribute> <.5> followed by a sequence of graphical primitives that are contained in the segment.

Intensity, line type and character size commands (just like those sent from SP to UP) are imbedded in the sequence of primitives whenever necessary in order to specify the segment exactly.

Finally, at the end of the sequence of primitives, a <seg.close> is sent if the segment was unposted, or a <seg.post> <\*seg.name\*> is sent if the segment was posted.

### III.12. Positioned Text

This section describes protocol for displaying positioned text on the screen. The positioned text facilities are intended to cater for the needs of display-oriented text editors, like NLS. The SP can instruct the UP to create "text windows" on the screen: a text window is a rectangular area in which a series of "lines" of text can be displayed. Protocol commands from the SP permit characters within a line to be changed, permit lines to be moved, and the like. All changes to the text windows specified by positioned text protocol take effect immediately.

The same mechanisms for dealing with positioned text can be used to provide "teletype simulation" for TELNET-like dialog with the server host, although this is entirely up to the UP implementation.

The commands for creating and destroying text windows are

```
<pTEXT.open> <*pTEXT.name*> <*count*> <*x.s.coord*> <*y.s.coord*>
          <*x.s.coord*> <*y.s.coord*> <*count*>
```

```
<pTEXT.kill> <*pTEXT.name*>
```

The open command creates a text window, specifies a unique name for it, the number of lines it is to contain (the first <\*count\*>), the coordinates of the lower left and upper right corners of the window in screen coordinates, and the (discrete) character size to use for all characters in the window (the second <\*count\*>). If the name specified in the open command already exists, the old window is destroyed.

The kill command destroys the named text window. If no such window exists, the command is ignored.

A number of attributes of a text window can be set with the optional command

```
<pTEXT.set> <*pTEXT.name*> <*pTEXT.flags*>
```

The <\*pTEXT.flags\*> byte is simply a <\*small.integer\*> of flag bits:

[‘001] Accept unescorted characters ("teletype simulation"). If this bit is on, any characters transmitted by the server host as part of TELNET protocol should be displayed in this text window. They are added "at the end," with normal ASCII conventions about format characters. Optional.

[‘002] Automatically scroll when text window fills up. This is probably only useful in conjunction with teletype simulation. Optional.

[‘004] Wrap long lines around to a new line. This too is most useful in conjunction with teletype simulation. A wrapped string counts as a new string. Optional.

[‘010] Make this text window visible. If the bit is off, the text window remains (and may be edited) but is not displayed. Optional.

[‘020] Make this text window "hit sensitive." That is, the input facilities for pointing can be applied to the window. Optional.

A default setting of <ptext.flags> to ‘010 is performed when a text window is first created.

There are several commands for changing the contents of strings in a text window. Strings within a text window are identified by a number (0 to n-1 where n is the number of strings in the window and the topmost string in the window is numbered 0). A <string.number\*> is simply the number of the string, in <count\*> format. Each string has an implicit length associated with it (internally in the UP); characters in a string are referred to by position (numbered starting at 1). Strings will normally contain standard ASCII printing characters.

The commands that modify strings are:

<ptext.scroll.up> <ptext.name\*> <string.number\*> <string.number\*>

Within a text window, scroll lines from the first string number to the second string number up one line, i.e. if the two string numbers specified are x and y, replace line x with x+1, x+1 with x+2, ... y-1 with y, and y with a null string.

<ptext.scroll.down> <ptext.name\*> <string.number\*> <string.number\*>

Similar, but scroll down.

<ptext.move> <ptext.name\*> <string.number\*> <ptext.name\*> <string.number\*>

This command causes the first string (specified by the text window name and string number) to be moved to the second string location. The first string is replaced with a null string.

<ptext.edit> <ptext.name\*> <string.number\*> <pos1\*> <pos2\*> <text\*>

This is the main command for editing strings. The arguments <pos1\*> and <pos2\*> are character positions within the specified string (in <count\*> format). The effect of the edit command is to replace the substring beginning with character pos1 and ending with character pos2-1 with the specified text string. If pos1 = pos2, this simply means that the string is inserted before the pos1 th character. There are two classes of illegal substrings that can be specified with pos1 and pos2:

`pos1 > length of string.` In this case, the text string is appended to the end of the present string.

`pos2 > (length of string)+1.` In this case, the command has the same effect as if `pos2` were exactly `(length of string)+1`.

Note that the text specified may be a null string -- hence if `pos1=1`, `pos2=large number` (e.g. 127), and `text=null`, the entire specified string is replaced by the null string.

**<ptext.modify>**    `<*ptext.name*>`    `<*string.number*>`    `<*pos1*>`    `<*pos2*>`  
`<*ptext.feature*>`

This optional command is used to select a substring that is to receive special treatment (e.g. highlighting, or blanking). It can also be used to make individual lines visible and later invisible. The values of `<*ptext.feature*>` are:

`<ptext.highlight.on>`  
`<ptext.highlight.off>`  
`<ptext.visible.on>`  
`<ptext.visible.off>`

Whenever a substring is inserted with the `<ptext.edit>` command, the default (`<ptext.highlight.off>`, `<ptext.visible.on>`) is established.

**<ptext.remote.edit>** `<*ptext.name*>` `<*string.number*>`

This optional command is an *ad hoc* attempt to allow UP's to implement various local line-editing regimens, without requiring interaction with the SH or SP. This command specifies a string in a positioned text window that is to be "edited" by the user. When the editing is finished, the line is returned to the SP via the TELNET connection. (This solution is unsatisfactory for several reasons, but it seems necessary to offer such a capability.)

(There have been some suggestions that we try to devise a subset of these facilities that could be implemented on some kinds of text terminal with no external character memory. If the terminal has the ability to insert and delete characters at will, and to do the scrolling functions listed above, then the only troublesome commands are the `<ptext.move>` command, which requires reading characters from an arbitrary line, and the `<ptext.modify>` command.)

### III.13. Input Facilities

This section describes a set of input facilities for the graphics protocol. They have been kept very simple to allow simple interaction with graphics programs without tremendous complexity. Specific interactive requirements may necessitate special-purpose protocols. The input facilities are optional: many graphics application programs need only keyboard facilities provided by TELNET.

Many details concerning provision of input facilities are left to the designers of the UP. The main reason for this approach stems from vast differences among operating systems and input device hardware -- no detailed set of specifications could be met by any one site. For example, an operating system may use some

"thinning" algorithm when passing tablet coordinates to a program in the system; different operating systems will surely have different conventions. In addition, individual (human) users may prefer slightly different styles of interactions; some of the stylistic flexibility that the protocol leaves open to the UP can be exploited by the user. However, the protocol does specify the purpose of each kind of input device and event. The UP should abide by the spirit of these descriptions, although the details may vary.

The Input provisions provide mechanisms for the server to:

1. Read the state of a device on demand.
2. Use an interactive technique, called an event.

### *III.14. Reading the State of Input Devices*

This section describes facilities for reading the state of any input device available at the UH.

The repertoire of available input devices is reported by the UP to the SP in the inquiry response. This list includes, for each device that is available, a one-byte "index" that uniquely identifies the device. This index, referred to as an <\*input.device.number\*>, is used by the SP to request measurement of the state of the device.

The protocol provides for the following five types of devices:

#### Coordinate Device

The state of a coordinate device is a pair of coordinates in the screen coordinate system. These coordinates could be derived from a tablet and stylus, from a light pen, from two knobs, from a keyboard (by typing in values, or using a keyboard-propelled cursor) or whatever.

#### Linear Device

The purpose of linear devices, such as knobs, is to provide a fraction in the range 0 to 1 to the SP. The UP may, if it wishes, provide some "echoing" of the current knob value, such as a changing number on the screen.

#### Key Device

The purpose of keys (or buttons) is to provide "function button" information to the SP. A key may be a single button (on/off) or a collection, arranged to form a number of possible code combinations (e.g., NLS keyset). The state of the key is a code describing the state of the key (bit=0 for normal position, bit=1 for depressed position).

#### Keyboard Device

A keyboard device provides a way of typing characters in the standard ASCII set. The state of the keyboard is the ASCII character code for the key most recently struck, or zero if the character has already been reported. (Note that the "state" of a keyboard is somewhat nonsensical -- we are more interested in the "events" caused by a keyboard. The keyboard is listed here for completeness.)

**Time**

If the UP provides a "time" input device, then it is willing to report to the SP a measurement of the current time. The protocol makes no tight specifications about how time is measured, except that it should be counted up once every 10 to 100 milliseconds. The UP may choose to count time somewhat inaccurately (e.g., by counting in an idle loop).

The SP can request that the state of a number of input devices be read and reported back with the optional command:

`<input.report> <*count*> <*input.device.number*> ...`

which specifies a list of devices whose states are to be measured as nearly simultaneously as possible and returned to the SP in the format:

`<input.report> <*device.report.sequence*>`

where `<*device.report.sequence*>` is a list:

`<*count*> <*input.device.report*> ...`

Each class of device has a canonical reporting format, `<*input.device.report*>`. The following paragraphs describe the format of `<*input.device.report*>`.

**Keyboard Device:** `<*input.device.number*> <*count*>`

The character code of the last key struck is returned in `<*count*>` format. Zero is returned if no key has been struck since the last report.

**Key Device:** `<*input.device.number*> <*count*>`

This report simply specifies the current value of the code of the key device (0 to n-1 where n is the number of states of the device).

**Linear Device:** `<*input.device.number*> <*large.fraction*>`

This simply reports the reading of the device as a fraction of its maximum excursion.

**Coordinate Device:** `<*input.device.number*> <*x.s.coord*> <*y.s.coord*> <*sw*>`

This report gives the current coordinates of the input device, and an indication of whether the "pen switch" (if it exists) is depressed (`<*sw*> = <on>`) or not (`<*sw*> = <off>`).

**Time Device:** `<*input.device.number*> <*time*>`

The time report specifies the current time. The format of `<*time*>` is `<*large.integer*>`. Note that time is recorded modulo  $2^{16}$ .

**III.15. Input Events**

The protocol provides a set of facilities for reporting to the SP the results of several kinds of input events initiated by the user. This is in contrast to the

"report-on-demand" facilities of the previous section. In particular, the protocol associates with several of the events a particular kind of interactive technique, often involving local feedback.

#### Keyboard Event

A keyboard event occurs when a key of a "keyboard device" is struck. The "report" for this event is the ASCII code of the key struck.

#### Key Event

Key events cause reports to be sent to the SP when the user manipulates a key device. A key event can be reported when the key is depressed or when it is released. (The NLS keyset is almost impossible to use unless key values are reported when a key is released.) The "report" is identical to reporting the state of a key, as described above.

#### Linear Event

The definition of this event, which can only be provided by "linear devices," is left to the UP. It might occur if the user changes the reading of a knob more than a certain threshold amount, or whatever.

#### Positioning Event

A coordinate device is used to specify a particular position or coordinate pair. For example, the user might briefly depress a tablet stylus and thus specify a position. The details of how the positioning event is caused are left up to the UP.

#### Pointing Event

A coordinate device is used to identify some segment, figure (structured picture definitions) or portion of positioned text that is currently displayed and that has been made "hit sensitive." The user can thus point at something on the screen. The SP expects to be told the name of the thing pointed at and the coordinates where the "hit" occurred. The UP can use a number of devices and techniques to provide these features (e.g. light pen, tablet and stylus with a hardware or software comparator). The details of when the hit is caused are left to the UP. The size of the "window" used to search for the hit is also left up to the UP (the human user may want to control this). As a local option, the UP may want to highlight in some way the segment or character that is pointed at. This identification can be removed when pointing is disabled or when the user is no longer pointing at the object (after a possible time lag). These details are up to the UP.

#### Stroke Event

A coordinate device is used to "draw" a stroke. The UP shows on the screen a track of ink left behind the coordinate positions, and reports to the SP the coordinates of points along the stroke. The details about when a stroke is terminated (and hence when the inking event is caused) are left to the UP. One common convention is to begin a stroke when the stylus pen switch is depressed, to continue recording points at some rate as long as the switch stays closed, and to terminate the stroke when the switch opens.

### Multiple Stroke Event

This event is similar to the stroke event, but is used to record a sequence of several strokes. This is useful for on-line character recognition. The usual technique is to assume the user is finished with a collection of strokes when a period of, say, 0.5 second has elapsed since the completion of the last stroke.

### Dragging Event

A coordinate device is used to move figures around on the display screen without requiring intervention from the SP. The idea is to attach a segment to the coordinates delivered from a coordinate input device, such as a tablet stylus. This interaction is not possible on many kinds of displays (e.g. storage tubes); UP's driving these terminals will not be able to provide dragging. On many refresh displays, an implementation of dragging is very simple: each segment can be defined as an absolute position, followed by relative motions (i.e. relative vectors, and relative invisible motions). In addition as the segment is recorded in the UP, we record the maximum and minimum values of x and y that the beam visits. In order to drag the segment about, we receive coordinates from the input device, check against the x and y maxima and minima to be sure that the new coordinate position will not cause any portion of the segment to go off the edge of the screen, and if not, the coordinates of the initial position instruction are replaced by the tablet coordinates (this check is only necessary for displays that cannot tolerate vectors or text that go off the screen). In order to be dragged, a segment must have been created with the <set.position.attribute> attribute specified (thus the UP can generate the segment of the display file as described above).

### Pendown, Penup Events

These are special cases of positioning events which may be meaningful in certain applications. The pendown event is caused when a stylus is pressed onto a tablet surface; the penup event when it is raised.

*If any of the positioning, pointing, stroke, multiple stroke, dragging, penup or pendown events are enabled, the UP should cause a tracking dot (or some form of positional feedback) to appear on the screen.*

### III.16. Enabling Events

The repertoire of Input events is reported by the UP to the SP in the inquiry response. This list includes a one-byte index, the <\*input.event.number\*>, that uniquely identifies each event implemented by the UP. If, for example, a pointing event can be caused by a light pen or by a stylus device, then two separate <\*input.event.number\*>s are assigned, one for the corresponding event on each device.

The SP may send commands that enable and disable input events. If an event is not enabled, the UP can ignore the corresponding device (i.e. need not buffer events that occur on an un-enabled device). (As a local option, the UP may send characters typed on an unenabled keyboard through the TELNET connection as "unescorted characters.")

When the SP enables for an event, it specifies the event being enabled, the conditions under which the event will be disabled, and what is to be reported when the event occurs. The command is:

```
<input.enable> <*input.event.number*>
    <*input.disable.condition*>
        <*input.report.sequence*>
```

The <\*input.disable.condition\*> is one of:

- |                   |   |
|-------------------|---|
| <never>           | Event remains enabled (until further notice from SP). |
| <with.this.event> | Disable this event when this event occurs.            |
| <with.any.event>  | Disable this event when the next event occurs.        |

When an enabled event occurs, the UP sends to the SP an <\*input.event.report\*> that describes the results of the event that occurred. In addition, the application program may desire that the state of various input devices be measured when the event occurs, and that these readings also be reported. When an event is enabled, the <\*input.report.sequence\*> specifies an ordered list of devices whose state should be measured:

```
<*count*> <*input.device.number*> ...
```

The UP should save the report list and associate it with the event that is enabled with this command. When the event occurs, the report list is used to compose a message for the SP that describes the event. (This is the mechanism used to report the time at which an event occurs.)

The dragging event requires an additional argument, the <\*seg.name\*> of the segment that is to be dragged. This is treated as a special case, and is set (before enabling) with the command:

```
<input.drag.set> <*input.event.number*> <*seg.name*>
```

An event may be explicitly disabled by

```
<input.disable> <*input.event.number*>
```

(If stroke collection is disabled, the ink is cleared. If pointing is disabled, any highlighting for the object pointed at can be cleared.)

The entire input system is cleared and all events disabled with the command

```
<input.reset.system>
```

### III.17. Event Reports

When an input event occurs, an event report is sent from the UP to the SP. The form of an event report is:

```
<input.report> <*input.event.report*> <*input.report.sequence*>
```

The <input.report.sequence\*> is defined above, and is the collection of state measurements made on various input devices when the event occurred. The <input.event.report\*> has a canonical form for each event class as follows (unless otherwise noted, the format is the same as the corresponding <input.device.report\*>):

**Keyboard Event:** <input.event.number\*> <count\*>

**Key Event:** <input.event.number\*> <count\*>

**Linear Event:** <input.event.number\*> <large.fraction\*>

**Positioning Event:** <input.event.number\*> <x.s.coord\*> <y.s.coord\*>

**Pendown Event:** <input.event.number\*> <x.s.coord\*> <y.s.coord\*>

**Penup Event:** <input.event.number\*> <x.s.coord\*> <y.s.coord\*>

These last three reports simply specify a coordinate pair, in the screen coordinate system.

**Pointing Event:** <input.event.number\*> <hit\*>

This report varies with different kinds of things that are hit. A <hit\*> is one of two things:

```
<segment.hit> <seg.name*> <x.s.coord*> <y.s.coord*>
<ptext.hit> <ptext.name*> <string.number*> <pos1*>
```

The first specifies the name of the entity that was pointed to. The last specifies the name of the text window, the string number and character position within the string that was identified.

**Stroke Event:** <input.event.number\*> <timed\*> <stroke\*>

There are two ways of reporting stroke information: with and without times associated with each point. The SP requests that time be reported by including the "time" device in the <input.report.sequence\*> when enabling the stroke event. The UP will then record times if it can.

The stroke report contains an indication of whether timing information is associated with each coordinate pair. If <timed\*> is <off>, a <stroke\*> is:

```
<count*> <x.s.coord*> <y.s.coord*> ...
```

where <count\*> is the number of coordinate pairs that follow. If <timed\*> is <on>, a stroke is:

```
<count*> <x.s.coord*> <y.s.coord*> <time*> ...
```

In other words, each coordinate pair has a time associated with it as well.

**Multiple Stroke Event:** <input.event.number\*> <timed\*> <count\*> <stroke\*> ...

This report is very similar to the stroke report, but has a provision for listing several strokes. The <count\*> is the number of strokes reported.

**Dragging Event: <\*input.event.number\*> <\*x.s.coord\*> <\*y.s.coord\*>**

This report simply specifies the coordinate pair that replaces the original home (i.e. first 'move' instruction) of the dragged segment.

**III.18. Inquiry**

The UP can transmit to the SP a number of bytes that describe the terminal serviced by the UP. This information is constant (so that a UP implementation does not have to compute it each time), and is usually requested by the SP at the beginning of a graphics session.

The SP can ask for the inquiry response by transmitting to the UP the command:

<Inquire>

The UP then transmits to the SP the sequence

<Inquire.response> <\*count\*> <\*response.phrase\*> ...

This response includes a count of the number of response "phrases" that follow, and the response phrases, in any order. A <\*response.phrase\*> is

<\*response.tag\*> <\*count\*> <\*response.value\*>

The count is the number of bytes in this particular <\*response.value\*>. The reason for this organization is to make the inquiry responses open-ended: the SP can ignore <\*response.value\*>s it cannot interpret.

In the description below, the tags and values for each kind of information are specified. If a default is specified, it may be assumed if the inquiry response does not include a phrase that overrides the default.

**UP Features****1. What protocol commands are implemented in the UP?**

Tag: <iImplemented.commands>

Value: up to 32 bytes of bit mask

The i<sup>th</sup> bit of the mask is a 1 if command i is implemented (i ranges from 0 to 255). Since the <\*response.phrase\*> for this information contains a count of the number of bytes that comprise the response, it is not necessary to provide the entire 32 bytes in the response. In the present protocol, there are only 102 command codes assigned (0-101), so only 13 bytes are required to completely describe which commands are implemented. This information is mandatory in the response.

**Terminal Features****2. What is the screen coordinate system?**

Tag: <i.screen.coordinates>

Value: <\*x.left.lv\*> <\*y.bottom.lv\*> <\*x.right.lv\*> <\*y.top.lv\*> <\*count\*>

This specifies precisely the admissible values for the <\*x.s.coord\*> and <\*y.s.coord\*> sequences in the subsequent protocol. The <\*count\*> specifies how many bytes are used to make up a coordinate value (e.g., this would be 2 for displays that are addressed as 0 to 1023). Thus, the number of bytes in a <\* .s.coord\*> is fixed by the UP.

The four <\* .lv\*> constructs that follow are examples of the screen coordinate system. Each <\* .lv\*> is a 4-byte sequence that specifies a 32-bit signed two's complement number. The four constructs thus specify the left, bottom, right, and top limits of the addressing space of the terminal.

*Example:* An IMLAC requires two bytes per coordinate; the lower-left corner of the screen is (0,0) and the upper right is (1023,1023). The 17 bytes of response are thus:

```
'000 '000 '000 '000 ;;x left
'000 '000 '000 '000 ;;y bottom
'000 '000 '003 '377 ;;x right
'000 '000 '003 '377 ;;y top
      '002 ;;number of bytes in <* .s.coord*>
```

As an example of the computation of a screen coordinate, suppose that we wish to compute the x screen coordinate of a spot that is the fraction  $f$  of the width of the screen ( $f=0$  is at the left edge,  $f=1$  at the right). Compute  $s = (<*x.right.lv*>-<*x.left.lv*>)*f + <*x.left.lv*>$ . Then transmit to the UP the low-order  $n$  bytes of the result, where  $n$  is the value of <\*count\*> in the response.

This response is mandatory.

### 3. What is the screen size?

Tag: <i.screen.size>

Value: <\*text\*> <\*text\*>

The two text strings specify, in decimal format (e.g. 126.43), the x and y dimensions of the screen in centimeters. Note that this format is chosen so that roll plotters can work effectively: they might choose a huge screen coordinate system, and specify a long dimension as well, e.g. 1000 centimeters.

### 4. How many screens are there?

Tag: <i.screen.number>

Value: <\*count\*>

Default is 1.

### 5. What is the device name?

Tag: <i.terminal.name>

Value: <\*text\*> <\*text\*>

Two strings are returned: the first is some form of manufacturer's name for the terminal (e.g. "IBM 2250"). The second is a string that can uniquely identify the terminal at the user site (e.g. "Terminal in room 34."). The protocol makes no specific format requirements on these strings -- they are for information only.

### 6. What is the terminal type?

Tag: <i.terminal.type>

Value: either <storage.terminal>, <storage.with.selective.erase>, <refresh.calligraphic> or <refresh.video>

### 7. How many resolvable intensity values are there?

Tag: <i.intensities>

Value: <\*count\*>

If the <\*count\*> value is n, then the permissible values as arguments to

the intensity-setting functions are 0 (no intensity) through n-1 (maximum intensity). Default n=2.

8. How many different line types are there?

Tag: <i.line.type>

Value: <\*count\*>

If the <\*count\*> value is n, then the permissible values as arguments to the type-setting functions are 0 (solid) through n-1. Default n=1.

9. What characters can be displayed on the terminal?

Tag: <i.characters>

Value: 32 bytes

Each of the 128 ASCII characters has a 2-bit code for it. The codes are 00 (cannot display), 01 (can display exactly), 10 (can transliterate, e.g. lower case to upper case, or tab to spaces), 11 (this is some visible character, but not ASCII). Default: 64 character ASCII.

10. What character orientations are there?

Tag: <i.character.orientations>

Value: <\*count\*> <\*large.fraction\*> ...

This response returns a list of available discrete orientations. Each fraction represents an angle (in range 0 to  $2\pi$ ) that is available (e.g. 0=normal horizontal; 1/4 is vertical running upward, etc.). If the <\*count\*> is n, then indices passed to the <set.character.orientation.discrete> command are in the range 0 to n-1. Default n=1, and the corresponding direction is 0.

11. What are the character sizes?

Tag: <i.character.size>

Value: <\*count\*> <\*char.size.description\*> ...

If <\*count\*> is n, then there are n discrete character sizes available, and the arguments to <set.character.size.discrete> should range from 0 to n-1. The character size descriptions are (as nearly as possible) in order of increasing size.

12. What input devices are available?

Tag: <i.available.input.device>

Value: <\*input.device.number\*> <\*input.device.description\*>

<\*events.provided\*> <\*text\*>

This response phrase specifies the identity of one input device (they are broken out so that the SP can skip over individual ones whose format it cannot interpret). The <\*input.device.number\*> is one byte that is to be used by the SP to reference the device. The <\*text\*> is a text string for human consumption that describes the device. (This string might be used by the SP to engage in a dialog with the user, asking him which devices to use for what function.) The <\*input.device.description\*> is one of:

```

<device.keyboard>
<device.key> <*count*>
<device.linear>
<device.coordinate>
<device.time> <*count*>

```

The key device gives, in <\*count\*>, the number of states the key can assume (e.g., an NLS keyset can assume 32 states). The time device gives, in <\*count\*>, the approximate number of milliseconds that elapse between increments to the time counter.

Each device also lists the kinds of events it can provide, and the <\*input.event.number\*>s used to reference the events. The reason for providing unique <\*input.event.number\*>s for each (device,event) pair is so that more than one device can provide similar interactive techniques. The <\*events.provided\*> is:

```
<*count*> <*input.event.number*> <*event.description*> ...
```

The count is the number of different events this device can provide. Each event is specified by its unique index and the <\*event.description\*>, which is one of:

```
<event.keyboard>
<event.key>
<event.linear>
<event.positioning>
<event.pointing>
<event.stroke>
<event.multiple.stroke>
<event.dragging>
<event.pendown>
<event.penup>
```

### III.19. Miscellaneous

This section describes a collection of miscellaneous commands provided in the protocol.

#### The sequence

```
<escape.protocol> <*text*>
```

is a way for the SP to transmit device-dependent information to the UP or from the UP to the SP. The protocol makes no provision for the format or encoding of the text string.

#### The command from SP to UP

```
<synchronize> <*count*>
```

causes the UP to respond

```
<synchronize> <*count*>
```

This provides a method of synchronizing things if absolutely necessary (e.g. a way of knowing whether a certain input event was caused before or after the display was changed with an "end batch of updates" command).

#### The command

```
<reset>
```

causes the UP to reset itself to a point right after the initial connection protocol has been completed and the connections opened. During early stages of debugging server and user software, this will doubtless be very useful.

Error conditions detected in the UP may be handled in several ways (the protocol makes no precise requirements):

1. Ignore them, or have the UP try to continue with as little damage as possible. Even severe errors should not crash the UP, since its operation is essential to permit the user to communicate with the SH and the application program.
2. Inform the SP of the error detected.
3. Inform the user (locally) of the error detected.

As a general strategy, the UP should probably try these in order. Certainly the UP should attempt to deal adequately with all errors (particularly such things as running out of display buffer space) so as to minimize the chances of a user losing a session's work.

Experience with common errors is probably needed before a useful error-management scheme can be devised. For this reason, we provide a mechanism for the UP to report to the SP any error conditions it detects (in free text format) and vice-versa. Thus system programmers at each end can, by saving and later examining error messages, keep track of major sources of error. The error report is

<error.string> <\*text\*>

In some networks, it may be necessary to establish synchronization of sender and receiver of protocol. For example, if a message is lost in the network, the UP may start interpreting operand bytes as if they were command codes. Since this is not a serious problem in the ARPA network, the present protocol does not enforce a synchronization mechanism. The following scheme is believed to be adequate, and will be instituted if network reliability is a problem:

We shall define a synchronization command, called GS, that has a code different from that of any protocol command. Whenever a data byte that is equal to GS is transmitted, it must be doubled (i.e., it is transmitted as GS, GS). A single GS may precede any protocol command code. Thus, whenever a receiver encounters a single GS, it knows that the next byte is a protocol command, and not an operand. The sender may transmit a GS preceding any command byte. It may choose to transmit as many or as few of these as seem appropriate.

## SECTION IV

### *Implementation Suggestions*

The details of the protocol may seem prohibitively forbidding. This section attempts to dispel that notion and to make it all appear so simple that standard mortals can implement it.

The op-code definitions (see following section) group the commands into four groups:

#### Mandatory

- Group 1 -- Transformed Format
- Group 2 -- Positioned Text
- Group 3 -- Input

Once the mandatory functions are implemented, any combination of the remaining three groups and the various optional commands may be implemented. For example, a UP that implements the Mandatory functions and Group 1 will be very useful indeed: many curve-plotting and mathematics packages that wish to do graphic output but are content with teletype-like input (provided by TELNET) can now be used.

An implementation of Mandatory and Group 2 would be adequate for simple page-oriented text editors; addition of Group 3 permits NLS and other more interactive systems to work. (The only combination that is probably not meaningful is just Mandatory and Group 3.)

Since many of the features of the protocol are optional, it is likely that many implementations will not include many of the options. *There is no stigma associated with omitting optional portions.*

In order to distribute information about sites that have implemented server or user programs that conform to the protocol, we would like to establish an informal "clearing-house" for such information. Those with information to give or request should address:

Robert F. Sproull  
Xerox Palo Alto Research Center  
3180 Porter Drive  
Palo Alto, Calif. 94304

or

SPROULL@PARC-MAXC (ARPA network mail)

Especially welcome is information about implementations of the protocol that can be offered to others (e.g., if someone writes an SP facility for INTERLISP, or a UP for an IMLAC).

## SECTION V

*Op-Code Assignments and Options*

This section assigns a number for each of the protocol command bytes described in section III, and tries to indicate what is optional and what is not.

The options column describes the conditions under which the UP should implement the command. M stands for "mandatory," and O for "optional." M-1 means mandatory if any commands in group 1 are implemented, i.e. If any M-1 command is implemented, then all M-1 commands must be. O-1 means optional if M-1 commands are implemented; otherwise not implemented.

The op-codes are grouped so that they may be decoded without requiring a full dispatch table if entire groups are unimplemented. The high-order three bits of the op code are the group number, the remaining five bits are the command within the group.

## Inquiry Commands

<inquire>	1	M
<inquire.response>	2	M

## General Commands

<escape.protocol>	3	O
<synchronize>	4	O
<reset>	5	O
<error.string>	6	O

## Transformed Format Commands (Group 1)

<seg.open>	32	M-1
<seg.close>	33	M-1
<seg.post>	34	M-1
<seg.unpost>	35	M-1
<seg.kill>	36	M-1
<end.batch.of.updates>	37	M-1
<seg.append>	38	O-1
<seg.dot>	39	M-1
<seg.move>	40	M-1
<seg.draw>	41	M-1
<seg.text>	42	M-1
<set.intensity>	43	O-1
<set.type>	44	O-1
<set.character.orientation.discrete>	45	O-1
<set.character.orientation.continuous>	46	O-1
<set.character.size.discrete>	47	O-1
<set.character.size.continuous>	48	O-1
<change.attribute>	49	O-1
<set.highlight.attribute>	50	O-1
<set.hit.sensitivity.attribute>	51	O-1
<set.intensity.attribute>	52	O-1
<set.screen.select.attribute>	53	O-1
<set.position.attribute>	54	O-1

<seg.readback.seglist>	65	O-1
<seg.readback(seg)>	66	O-1

**Positioned Text Commands (Group 2)**

<ptext.open>	64	M-2
<ptext.kill>	65	M-2
<ptext.set>	66	O-2
<ptext.scroll.up>	67	O-2
<ptext.scroll.down>	68	O-2
<ptext.move>	69	O-2
<ptext.edit>	70	M-2
<ptext.modify>	71	O-2
<ptext.remote.edit>	72	O-2

**Input Commands (Group 3)**

<input.enable>	96	M-3
<input.disable>	97	M-3
<input.report>	98	O-3
<input.event>	99	M-3
<input.reset.system>	100	M-3
<input.drag.set>	101	O-3

**Other code assignments (these are *not* commands)**

<off>	0
<on>	1

**Positioned Text**

<ptext.visible.off>	0
<ptext.visible.on>	1
<ptext.highlight.off>	2
<ptext.highlight.on>	3

**Input facilities**

<device.keyboard>	0
<device.key>	1
<device.linear>	2
<device.coordinate>	3
<device.time>	4

<event.keyboard>	0
<event.key>	1
<event.linear>	2
<event.positioning>	5
<event.pointing>	6
<event.stroke>	7
<event.multiple.stroke>	8
<event.dragging>	9
<event.pendown>	10
<event.penup>	11

<never>	0
<with.this.event>	1
<with.any.event>	2

<segment.hit>	0
<ptext.hit>	1

Inquiry tag definitions		
<i.implemented.commands>	1	M
<i.screen.coordinates>	2	M
<i.screen.size>	3	O
<i.screen.number>	4	O
<i.terminal.name>	5	O
<i.terminal.type>	6	O
<i.intensities>	7	O
<i.line.type>	8	O
<i.characters>	9	O
<i.character.orientations>	10	O
<i.character.size>	11	O
<i.available.input.device>	12	O
<refresh.calligraphic>	0	
<storage.terminal>	1	
<storage.with.selective.erase>	2	
<refresh.video>	3	

## Appendix

### Outline of Structured Format Protocol

This appendix presents a preliminary design for a structured output format protocol. It is similar to the "groups and items" technique [N&S]. It caters primarily for high-performance displays that are capable of implementing transformations in hardware and of interpreting a structured display file. However, software processes can be used by a UP to simulate these facilities if the display does not have the capability.

#### *Display Structure*

A *display structure* consists of *figures*, each containing any number of *units*. There are two types of units, *primitive* units and *call* units. Primitive units contain drawing instructions and associated coordinates that may generate visible information on the display screen. Drawing instructions and coordinates can occur only in primitive units.

Call units give the display structure a subroutine capability. A call unit invokes the display of another figure. In other words, a call unit allows one figure to contain *instances* of other figures. As well as providing for subroutine-style control transfer, call units can be used to establish the parameters to be used in the display of the subfigure. For example, a call unit can be used to call a figure with a specified intensity setting and translation. On return from the called figure, these parameters are restored to their original values.

A figure is an ordered list of units which can be any mixture of primitive and call units. Each figure begins with a *header* and terminates with the *figure end unit*. The ordering of units within a figure does not affect the display produced, but, particularly in languages such as LISP, it may be convenient to have this ordering correspond to some application data structure ordering.

In order to understand how control passes through a structure, one can think of the display elements as follows: figures are subroutines and units are linked blocks of in-line code. When all of the units contained in a figure have been executed, the figure end unit returns control to wherever the figure was called from. A primitive unit contains line and character descriptions and a transfer to the next unit. A call unit contains a subroutine call to a subfigure and a transfer to the next unit in line. Figure 7 shows a typical display structure.

#### *Accessing Mechanisms*

Figures are referenced by user-assigned names. Units may be given names at the option of the user. No two figures can have the same name, and no two units within a figure can have the same name. However, units in separate figures can have identical names, and a unit can be "named after" (i.e., carry the same name as) a figure. Figure names are called *global* and unit names *local* to reflect the fact that a unit name only distinguishes between the units within a figure.

To reference a figure, one merely refers to it by name. To reference a unit within a figure, one supplies both the figure and unit names. It is this naming mechanism which makes it possible to make incremental changes in the display.

The display structure exists at the UH, rather than at the application program.

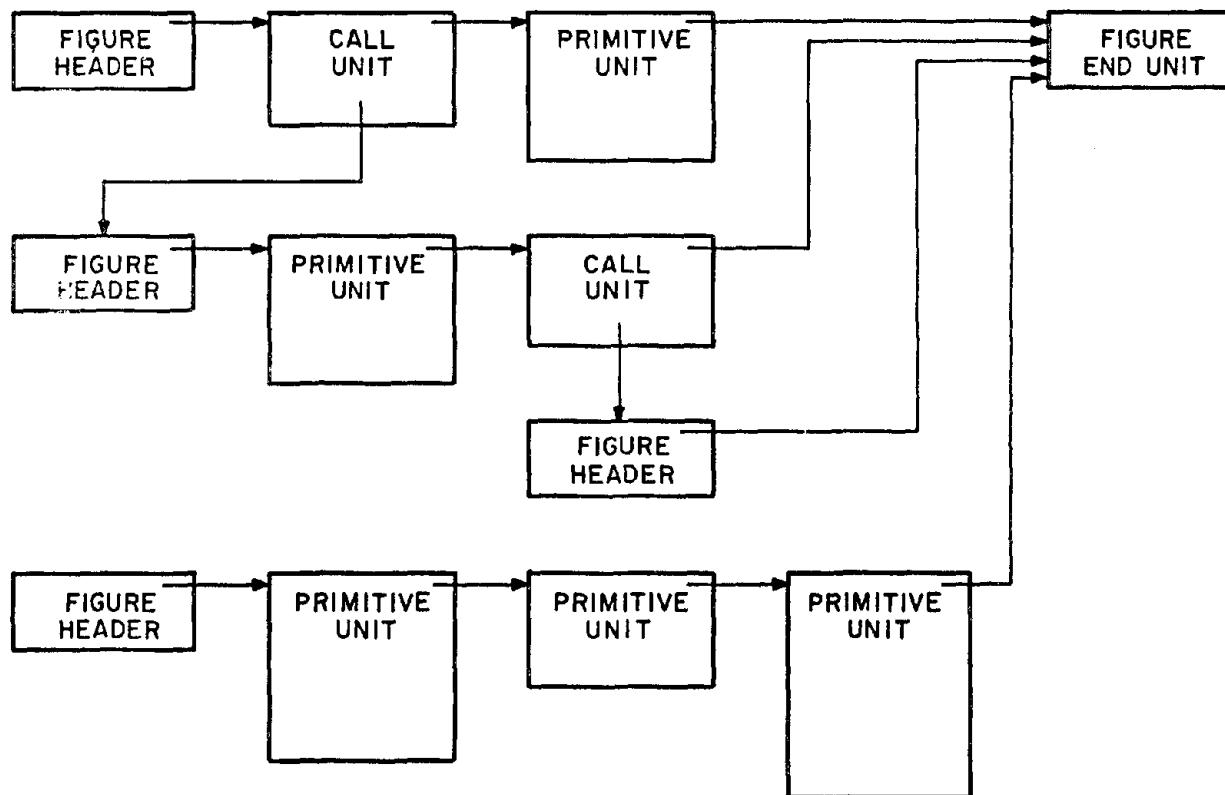


Figure 7: A Typical Display Structure

For figure and unit names to be useful, the application program should generate display names carefully to insure a one-to-one correspondence between display names and application program data structure parts.

#### *Primitive Unit*

A primitive unit can be used to draw any combination of lines and characters. More specifically, a primitive unit can consist of any number or combination of graphical primitives for drawing dots, lines and text (similar to the primitives for the transformed format, section III.5). Display coordinates are two's complement fractions of appropriate resolution centered about the point (0,0).

An option should be provided in the protocol to specify graphical primitives in a three-dimensional coordinate system.

#### *Call Unit*

Call units enable several similar picture parts to be described by the same figure. For example, the display of a circuit diagram can be constructed from a figure which is composed of the lines for a resistor and call units of this figure for each resistor in the display. All coordinate transformations and changes in intensity are specified as parameters of call units.

Master and instance rectangles are used to specify the clipping and scaling of

coordinates. A *master rectangle* is an area in the called figure which is to be mapped into an area specified by the *instance rectangle* in the calling figure (see Figure 8). Lines in the called figure which have coordinates outside the master rectangle are not displayed in the calling figure; lines in the called figure which cross the master rectangle are clipped when they are displayed in the calling figure.

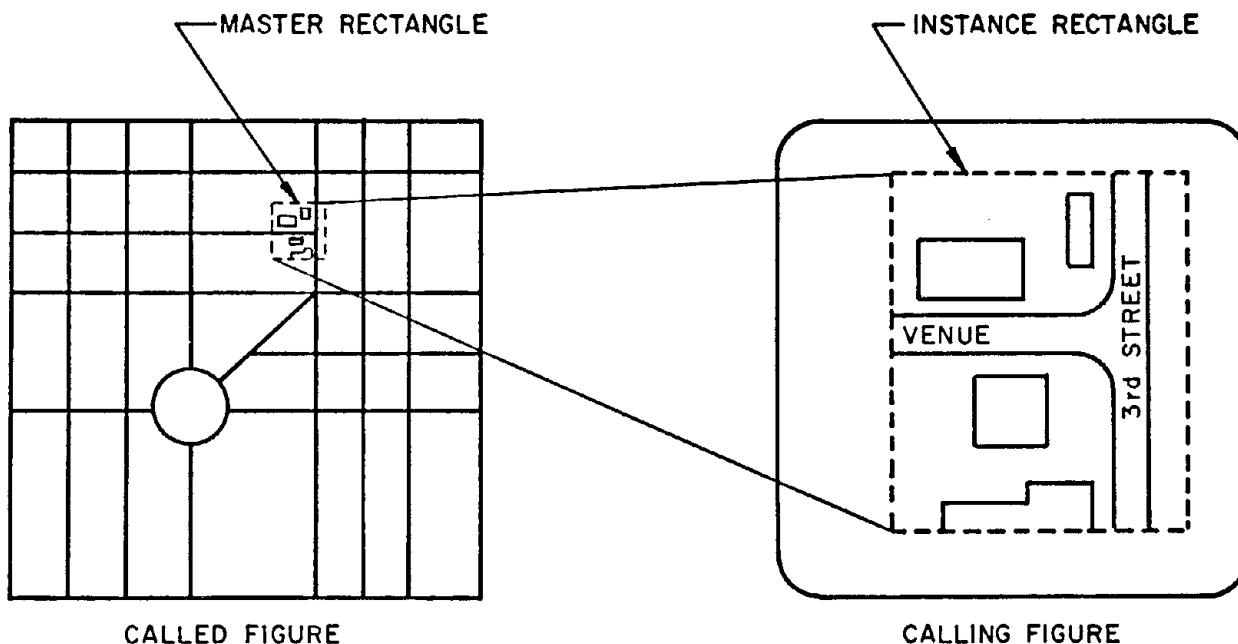


Figure 8: Clipping: Master and Instance Rectangles

Several ways of describing the master and instance rectangles of a call unit should be provided in the protocol:

1. If no master and instance rectangles are specified when creating a call unit, the coordinates of the called figure are not clipped or transformed but are displayed just as they appear in the called figure.
2. A *move call unit* has master and instance rectangles which are maximum in size and offset with respect to each other, thus "moving" all the coordinates of the called figure by a specified amount in the calling figure.
3. A *scale call unit* moves and scales the called figure with respect to the calling figure. Either the master rectangle or the instance rectangle is maximum in size, and the other rectangle is suitably smaller to achieve the desired scale.
4. Both the master and instance rectangles can be specified directly by the user. A rectangle is designated by its lower left-hand and upper right-hand corners.

A call unit also can specify an angle of rotation to be applied to the called figure. Display coordinates are rotated before they are scaled and clipped.

An option should be provided so that a call unit can specify a three-dimensional transformation. This allows displays of graphical primitives that have been specified in three dimensions. Ideally, the transformation should include the ability to specify a perspective view.

The Intensity value specified in a call unit is the absolute intensity level at which the called figure is to be displayed. If no intensity is specified, the intensity remains unchanged from that of the calling figure. The default intensity of the display is the scope's brightest intensity.

Many displays have blinking line and dashed line capabilities. A call unit can be used to change the line type to dashed and/or blinking for all the lines of the called figure. If a particular scope does not have the requested capability then the line type remains unchanged (analogous to transformed format, section III.8).

#### *Display Structure Construction and Modification*

A display structure is constructed by creating its units. When a unit is created, the figure containing the unit must be specified. If the figure does not exist, it is also created. In addition, if the figure called by a call unit does not exist, it is created. Thus, new figures are created implicitly by placing units in them or by calling them from some other figure.

A unit can be inserted in the structure in one of three ways:

1. It can be inserted at the end of a figure.
2. It can be inserted after a particular unit in a figure (where the figure header is an acceptable unit after which to insert).
3. It can replace a particular unit which already exists.

If a unit with the same local name as the new unit already exists in the figure, the old unit will be deleted as a result of the creation of the new unit.

The function "display figure" causes the specified figure to be displayed. The arguments to this function are similar to those of a call unit: a master rectangle is applied to the called figure, and mapped onto a "viewport," specified in the screen coordinate system. This call unit is distinguished from all others because it alone specifies a mapping from the large two's complement coordinate system of figures and units to the screen coordinate system (which may not have a square aspect ratio). At any one time, only a single figure can be displayed. However, this is not restrictive since the figure on display can call any or all other figures. As units are added to the displayed figure, they are displayed. An empty figure is created as a result of the "display figure" function if the figure does not already exist. The function "clear scope" removes all such displayed figures from view.

The protocol should provide several mechanisms to change a display structure incrementally. Individual units and figures can be deleted from the display structure, the names of figures and units can be changed, and units can be blanked and unblanked.

Three types of deletion operations may be performed:

1. A single unit can be deleted.

2. A figure can be cleared, an operation which deletes each of the units of a figure but retains the figure header.
3. A figure can be deleted, thereby deleting all the units of the figure, the figure header, and all call units which reference the figure.

In all of the deletion operations, there is no error generated if the object to be deleted does not exist.

A unit or figure can be given a new name. If a unit is given the same name as some other unit in the same figure, the unit originally carrying the name is eliminated. If a figure name is changed to that of some other figure, the figure that already had the name is eliminated, along with any calls to it.

A blanked unit is a unit which exists in the display structure but which is marked not to be displayed. Thus, the lines and characters of a blanked primitive unit are not drawn and the figure referenced by a blanked call unit is not processed. Blanking and unblanking a unit is more efficient than deleting and recreating it.

If the transformations are being interpreted in software, it is often desirable to make several changes to the display structure before repainting the scope with the updated picture. The "end batch of updates" command is used to cause a complete update to the visible display.

#### *Input Facilities*

Two of the input facilities of the protocol take on a different meaning in conjunction with a structured display file: dragging and pointing.

For pointing, the SP can make individual figures "hit sensitive." Then, if the pointing event is enabled and the user identifies an object visible on the screen, the event report cites the global name and local name (if any) of the unit "hit."

Dragging is accomplished by identifying a move call unit whose parameters are to be changed by coordinates delivered from a coordinate input device. (For simplicity, the input device coordinates are used directly as the offset of the called figure to the calling figure. Thus, if the coordinates of the calling figure are transformed in any way, the movement of the called figure will be related to, but not directly tied to the movement of the input device.)

R E F E R E N C E S

[N&S]

W.M. Newman and R.F. Sproull, *Principles of Interactive Computer Graphics*, McGraw Hill, 1973.

[10GR]

W.M. Newman and R.F. Sproull, "An Approach to Graphics System Design," *Proceedings of the IEEE*, April 1974.

[Omnigraph.brief]

R.F. Sproull, "Omnigraph -- Simple Terminal-Independent Graphics Software," Xerox PARC Report CSL-73-4.

[Omnigraph]

"PDP-10 Display Systems," available from Computer Center Branch, Division of Computer Research and Technology, National Institutes of Health, Bethesda, Maryland 20014.

[NIC 19933]

"Proposed Network Graphics Protocol," Network Information Center 19933. (Unfortunately, this is no longer available from the NIC.)

[NLS]

D.C. Engelbart and W.K. English, "A Research Center for Augmenting Human Intellect," FJCC 1968, p. 395.

I N D E X

<\*attribute\*> 23  
<\*char.size.description\*> 23  
<\*count\*> 15  
<\*device.report.sequence\*> 30  
<\*hit\*> 34  
<\*input.device.number\*> 29  
<\*input.device.report\*> 30  
<\*input.disable.condition\*> 33  
<\*input.event.number\*> 32  
<\*input.event.report\*> 34  
<\*input.report.sequence\*> 33  
<\*large.fraction\*> 15  
<\*large.integer\*> 15  
<\*ptext.feature\*> 28  
<\*ptext.flags\*> 26  
<\*ptext.name\*> 15  
<\*response.phrase\*> 35  
<\*response.tag\*> 35  
<\*response.value\*> 35  
<\*seg.name\*> 15  
<\*small.fraction\*> 15  
<\*small.integer\*> 15  
<\*string.number\*> 27  
<text\*> 15  
<time\*> 30  
<timed\*> 34  
<x.s.coord\*> 21, 35  
<y.s.coord\*> 21, 35  
<change.attribute> 23  
<end.batch.of.updates> 20  
<error.string> 39  
<escape.protocol> 38  
<input.disable> 33  
<input.draq.set> 33  
<input.enable> 33  
<input.report> 30, 33  
<input.reset.system> 33  
<inquire.response> 35  
<inquire> 35  
<never> 33  
<ptext.edit> 27  
<ptext.kill> 26  
<ptext.modify> 28  
<ptext.move> 27  
<ptext.open> 26  
<ptext.remote.edit> 28  
<ptext.scroll.down> 27  
<ptext.scroll.up> 27  
<ptext.set> 26  
<reset> 38  
<seg.append> 19

<seg.close> 19  
<seg.dot> 21  
<seg.draw> 21  
<seg.kill> 19  
<seg.move> 21  
<seg.open> 19  
<seg.post> 19  
<seg.readback.seg> 25  
<seg.readback.seglist> 25  
<seg.text> 21  
<seg.unpost> 19  
<set.character.orientation.continuous> 23  
<set.character.orientation.discrete> 22  
<set.character.size.continuous> 23  
<set.character.size.discrete> 23  
<set.highlight.attribute> 24  
<set.hit.sensitivity.attribute> 24  
<set.intensity.attribute> 24  
<set.intensity> 21  
<set.position.attribute> 25  
<set.screen.select.attribute> 25  
<set.type> 22  
<synchronize> 38  
<with.any.event> 33  
<with.this.event> 33



K. Harrenstien (SRI-KL)  
RFC 738, NIC 42218 (Oct. 31, 1977)

Time Server

This note describes the Time Server protocol, as currently implemented on ITS hosts (i.e. MIT-(AI/ML/MC/DMS)). The idea is to provide a site-independent, machine readable date and time in as efficient and swift a manner as possible; its motivation arises from the fact that not all systems have a date/time clock, and all are subject to occasional human or machine error. The use of time-servers makes it possible to quickly confirm or correct a system's idea of the time, by making a brief poll of several independent sites on the network.

In particular the network time server works as follows:

S: Listen on socket 37 (45 octal).

U: Connect to socket 37 [not ICP].

S: Send the time as a 32 bit binary number.

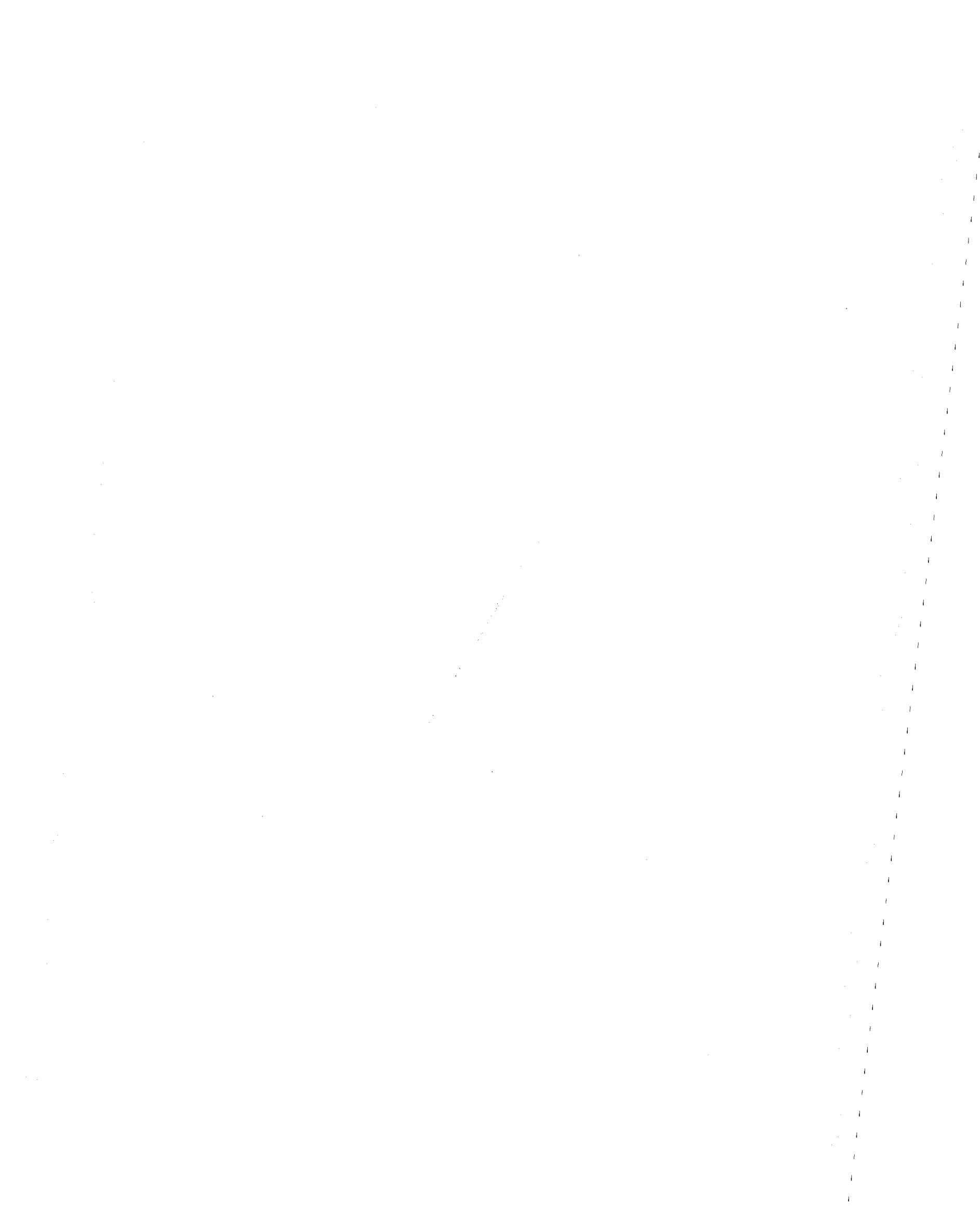
U: Close the connection.

S: Close the connection.

Note that this is not a normal ICP sequence. Rather, instead of sending a 32-bit socket number to reconnect to, the server returns a 32-bit time value and closes the connection. If the server is unable to determine the time at its site, it should either refuse the connection or close it without sending anything.

The time is the number of seconds since 0000 (midnight) 1 January 1900 GMT, such that the time 1 is 12:00:01 am on 1 January 1900 GMT; this base will serve until the year 2036. As a further example, the most recent leap year as of this writing began from the time 2,398,291,200 which corresponds to 0000 1 Jan 1976 GMT.

Preceding page blank



K. Harrenstien  
RFC 742, NIC 42758 (Dec. 30, 1977)

## NAME/FINGER

### Introduction

This note describes the Name/Finger protocol. This is a simple protocol which provides an interface to the Name and Finger programs at several network sites. These programs return a friendly, human-oriented status report on either the system at the moment or a particular person in depth. Currently only the SAIL (SU-AI), SRI (SRI-(KA/KL)), and ITS (MIT-(AI/ML/MC/DMS)) sites support this protocol, but there are other systems with similar programs that could easily be made servers; there is no required format and the protocol consists mostly of specifying a single "command line".

To use via the network:

ICP to socket 117 (oct), 79 (decimal) and establish two 8-bit connections.

Send a single "command line", ending with <CRLF>.

Receive information which will vary depending on the above line and the particular system. The server closes its connections as soon as this output is finished.

The command line:

Systems may differ in their interpretations of this line. However, the basic scheme is straightforward: if the line is null (i.e. just a <CRLF> is sent) then the server should return a "default" report which lists all people using the system at that moment. If on the other hand a user name is specified (e.g. FOO<CRLF>) then the response should concern only that particular user, whether logged in or not.

Both ITS and SAIL sites allow several names to be included on the line, separated by commas; but the syntax for some servers can be slightly more elaborate. For example, if "/W" (called the "Whois switch") also appears on the line given to an ITS server, much fuller descriptions are returned. The complete documentation may be found at any time in the files ".INFO.;NAME ORDER" on MIT-AI, "FINGER.LES[UP,DOC]" on SU-AI, and "<DOCUMENTATION>FINGER.DOC" on

---

Preceding page blank

Name/Finger  
RFC 742, NIC 42758 (Dec. 30, 1977)

SRI-KL, all freely accessible by FTP (with the exception of SRI-KL, where TOPS-20 requires the "anonymous" login convention).

Allowable "names" in the command line should of course include "user names" or "login names" as defined by the system, but it is also reasonable to understand last names or even full names as well. If a name is ambiguous, all possible derivations should be returned in some fashion; SAIL will simply list the possible names and no more, whereas an ITS server will furnish the full standard information for each possibility.

Response to null command line - "default" listing:

This is a request for a list of all online users, much like a TOPS-10 or TENEX "systat". To fulfill the basic intent of the Name/Finger programs, the returned list should include at least the full names of each user and the physical locations of their terminals insofar as they can be determined. Including the job name and idle time (number of minutes since last typein, or since last job activity) is also reasonable and useful. The appendix has examples which demonstrate how this information can be formatted.

Response to non-null command line - "name" listing:

For in-depth status of a specified user, there are two main cases. If the user is logged in, a line or two is returned in the same format as that for the "default" listing, but showing only that user. If not logged in, things become more interesting. Furnishing the full name and time of last logout is the expected thing to do, but there is also a "plan" feature, wherein a user may leave a short message that will be included in the response to such requests. This is easily implemented by (for example) having the program look for a specially named text file on the user's directory or some common area. See the examples for typical "plans".

Implementation miscellany:

Anyone wishing to implement such a server is encouraged to get in touch with the maintainers of NAME by sending a message to BUG-NAME @ MIT-AI; apart from offering advice and help, a list of all sites with such servers is kept there. It is also suggested that any existing programs performing similar functions locally (i.e. not as net servers) be extended to allow specification of other sites, or names at other sites. For example, on ITS systems one can say ":NAME<cr>" for a local default listing, or ":NAME @SAIL<cr>" for SAIL's default listing, or ":NAME Foo@MC<cr>" to ask MIT-MC about Foo's status, etc.

It should be noted that connecting directly to the server from a TIP or an equally narrow-minded TELNET-protocol user program can result in meaningless attempts at option negotiation being sent to the server, which will foul up the command line interpretation unless the server knows enough to filter out IAC's and perhaps even respond negatively (IAC WON'T) to all option commands received. This is a convenience but is not at all required, since normally the user side is just an extended NAME/FINGER type program.

And finally a little background:

The FINGER program at SAIL, written by Les Earnest, was the inspiration for the NAME program on ITS. Earl Killian at MIT and Brian Harvey at SAIL were jointly responsible for implementing the protocol just described, and Greg Hinchliffe has recently brought up a similar server for SRI-KA and SRI-KL.

Name/Finger  
Appendix -- Examples

EXAMPLES

NOTE: It is possible for some lines of the actual output to exceed 80 chars in length. The handling of such lines is of course dependant on the particular user program; in these examples, lines have been truncated to 72 chars for greater clarity.

Three examples with a null command line:

Site: MIT-AI

Command line:

-User-	--Full name--	Jobnam	Idle	TTY	-Console location-
XGP	O Xerox Graphics Printer	XGPSPL	T24	Datapoint	Near XGP (9TH)
FFM	U Steven J. Kudlak	HACTRN	T41	Net site	CMU-10A
KLH	+ Ken Harrenstien	F	T42	Net site	SRI-KL
<u>013</u>	- Not Logged In	HACTRN	1:26.T43	DSSR	UNIX x3-6048 (MIT-*)
CWH	U Carl W. Hoffman	E	4.T50	919	Very Small Data Bas*
CARL	A Carl Hewitt	HACTRN	5:03.T52	813	Hewitt x5873
APD	M Alexander Doohovskoy	XGP	1:52.T54	912	9th Floor Lounge x6*
JJK	T James Koschella	E	T55	824	Hollerbach, Levin, *
KEN	L Kenneth Kahn	E	T56	925	Moon (Tycho under) *

Site: SAIL

Command line:

Person	Job	Jobnam	Idle	Terminal				
DAN Dan Sleator	46	MACLSP		DM-3		150/1200	modem	415 49*
DEK Don Knuth	3	E	3.	tv-55	205	Library		
	20	PI	2	TV-55	205	Library		
ES Gene Salamin	44	SD MC		TV-40	223a	Farmwald		
JJ Jerryold Ginsparg	11	TELNET		DM-0		150/1200	modem	415 49*
JMC John McCarthy	1	FINGER	.	detached		McCarthy's house		
	12	E	2.	IML-15		Allen		
KRD Randy Davis	42	AID	7	TV-52	203			
LES Les Earnest	23	TEMPS	2.	DM-1		150/1200	modem	415 49*
ME Martin Frost	17	E	3	tv-46	220	Filman, Frost		
	31	E		TV-46	220	Filman, Frost		
PAM Paul Martin	9	E		TV-106	251C	King, Levy, Martin		
ROD Rod Brooks	37	MACLSP	3	TV-117	250C	Robinson		
RWG Bill Gosper	30	SD MC		TV-34	230e	Kant, McCune, Steinbe*		
				TV-67	213	Weyhrauch		
RWW Richard Weyhrauch	39	E		TV-42	214			
SYS system files	6	FINGER		PTY122		job 5 Arpanet site AI*		

Name/Finger  
Appendix - Examples

Site: SRI-KL

Command line:

Thursday, 15-Dec-77 01:21:24-PST System up 3 Days, 22:20:52 28 Jobs  
 Drum 0% Load avs 0.26 0.23 0.31 14 Act, 10 Idle, 8 Det

User	Personal Name	Job	Subsys	15m%	TTY	Room	Console	Location
BLEAN	Bob Blean	37	EXEC	0.0	41	K2007	Blean	
KLH	Ken Harrenstien	83	TELNET	1.6	12	J2023	Spaceport	
KREMERS	Jan Kremers	48	TECO	0.0	121	Dialup	326-7005 (300 Ba*	
MAINT	Digital Equipment	54	SNDMSG	0.5	43	K2035	Melling	
MCCLURG	Jim McClurg	40	EXEC	0.0	26	PKT		
MMCM	Michael McMahon	31	EXEC	1.5	122	Dialup	326-7006 (300 Ba*	
MOORE	J Moore	52	TV	0.2	124	Dialup	326-7008 (300 Ba*	
PATTIS	Richard Pattis	19	LISP	0.8	11	ARC		
PETERSO	Norman Peterson	33	EXEC	25:12	234		(RAND-TIP)	
STONE	Duane Stone	34	TELNET	3:51	240		(RADC-TIP)	
		27	EXEC	7:11	232		(SRI-KL)	
TORRES	Israel Torres	64	BSYS	0.0	76	K2079	TI by tape drives	
		68	EXEC	1:15	104	K2029	Operators' Office	

Name/Finger  
Appendix - Examples

Examples with names specified:

Site: MIT-AI  
Command line: klh

KLH + Ken Harrenstien      Last logout 10/16/77 13:02:11 No plan.

Site: MIT-MC  
Command line: cbf

CBF M Charles Frankston      Not logged in. Plan:  
I'll be visiting another planet til about December 15. If anyone  
wants to get a hold of me transmit on some fundamental wavelength  
(like the radius of the hydrogen atom).

Site: MIT-MC  
Command line: smith

BRIAN	A Brian C. Smith	Last logout 11/24/77 08:02:24	No plan.
DBS	T David B. Smith	Last logout 12/03/77 11:24:01	No plan.
BPS	T Byron Paul Smith	Not logged in.	No plan.
GRS	U Gary R. Smith	Last logout 12/12/77 18:43:19	No plan.
JOS	S Julius Orion III Smith	Last logout 11/29/77 06:18:18	No plan.
\$PETE	M PETER G. SMITH,	Not logged in.	No plan.
IAN	L Ian C. Smith	Not logged in.	No plan.
AJS	D Arnold J. Smith	Last logout 12/09/77 14:31:11	No plan.

Site: SU-AI  
Command line: smith

"SMITH" is ambiguous:

RS Bob Smith  
DAV Dave Smith  
JOS Julius Smith  
LCS Leland Smith

Name/Finger  
Appendix - Examples

Site: SU-AI

Command line: jbr

Person	Job	Jobnam	Idle	Line	Room	Location
JBR Jeff Rubin	16	COPY	27.	TV-43	222	Rubin
				TV-104	233	hand-eye table

Site: SU-AI

Command line: bh

Person	Last logout
BH Brian Harvey	22:49 on 14 Dec 1977. Plan: †008-Oct-77 2156 BH †Y12257 (1-Jul-78)

Weekdays during the day I'm usually unreachable; I'm either at S.F. State or at Benjamin Franklin JHS in San Francisco, but neither place is recommended for leaving messages. Evenings and weekends I'm generally home (55) 751-1762 unless I'm at SAIL. I log in daily from home.

Site: SRI-KL

Command line: greg

GREG (Greg Hinchliffe) is on the system:

Job	Subsys	#	Siz	Runtime	1m%	15m%	TTY	Room	Console	Location
62	EXEC	1	0	0:00:10.6		0.8	235		(SUMEX-AIM)	

Last login: Mon 12-Dec-77, 15:05, from SUMEX-AIM (Host #56.)  
GREG has no new mail, last read on Mon 12-Dec-77 15:10



Assigned Numbers  
RFC 739, NIC 42423 (Nov. 11, 1977)

J. Postel  
RFC 739, NIC 42423 (Nov. 11, 1977)

ASSIGNED NUMBERS

This Network Working Group Request for Comments documents the currently assigned values from several series of numbers used in network protocol implementations. This RFC will be updated periodically, and in any case current information can be obtained from Jon Postel. The assignment of numbers is also handled by Jon. If you are developing a protocol or application that will require the use of a link, socket, etc. please contact Jon to receive a number assignment.

Jon Postel  
USC - Information Sciences Institute  
4676 Admiralty Way  
Marina del Rey, California 90291

Phone: (213) 822-1511

ARPANET mail: POSTEL@ISIB

Most of the protocols mentioned here are documented in the RFC series of notes. The more prominent and more generally used are documented in the Protocol Handbook [1] prepared by the Network Information Center (NIC). In the lists that follow a bracketed number, e.g. [1], off to the right of the page indicates a reference to the protocol assigned that number.

Preceding page blank

Assigned Numbers

RFC 739, NIC 42423 (Nov. 11, 1977)

Assigned Link Numbers

ASSIGNED LINK NUMBERS

The word "link" here refers to a field in the original ARPANET Host/IMP interface leader. The link was originally defined as an 8 bit field. Some time after the ARPANET Host-to-Host (AHHP) protocol was defined and, by now, some time ago the definition of this field was changed to "Message-ID" and the length to 12 bits. The name link now refers to the high order 8 bits of this 12 bit message-id field. The low order 4 bits of the message-id field are to be zero unless specifically specified otherwise for the particular protocol used on that link. The Host/IMP interface is defined in BBN report 1822 [2].

Link Assignments:

Decimal	Octal	Description	References
0	0	AHHP Control Messages	[1,3]
1	1	Reserved	
2-71	2-107	AHHP Regular Messages	[1,3]
72-151	110-227	Reserved	
152	230	PARC Universal Protocol	
153	231	TIP Status Reporting	
154	232	TIP Accounting	
155-158	233-236	Internet Protocol	[35,36]
159-191	237-277	Measurements	[28]
192-195	300-303	Message Switching Protocol	[4,5]
196-255	304-377	Experimental Protocols	

Assigned Socket Numbers

ASSIGNED SOCKET NUMBERS

Sockets are used in the AHHP [1,3] to name the ends of logical connections which carry long term conversations. For the purpose of providing services to all callers an Initial Connection Procedure ICP [1,34] is used between the user process and the server process. This list specifies the socket used by the server process as its contact socket.

Socket Assignments:

General Assignments:

Decimal	Octal	Description
-----	-----	-----
0-63	0-77	Network Wide Standard Function
64-127	100-177	Hosts Specific Functions
128-223	200-337	Reserved for Future Use
224-255	340-377	Any Experimental Function

## Assigned Numbers

RFC 739, NIC 42423 (Nov. 11, 1977)

## Assigned Socket Numbers

## Specific Assignments:

Decimal	Octal	Description	References
<hr/>			
		Network Standard Functions	
1	1	Old Telnet	[6]
3	3	Old File Transfer	[7, 8, 9]
5	5	Remote Job Entry	[10]
7	7	Echo	[11]
9	11	Discard	[12]
11	13	Who is on or SYSTAT	
13	15	Date and Time	
15	17	Who is up or NETSTAT	
17	21	Short Text Message	
19	23	Character generator or TTYTST	[13]
21	25	New File Transfer	[1, 14, 15]
23	27	New Telnet	[1, 16, 17]
25	31	Distributed Programming System	[18, 19]
27	33	NSW User System w/COMPASS FE	[20]
29	35	MSG-3 ICP	[21]
31	37	MSG-3 Authentication	[21]
33	41	DPS ICP	[18, 19]
35	43	IO Station Spooler	
37	45	Time Server	[22]
39	47	NSW User System w/SRI FE	[20]
<hr/>			
		Host Specific Functions	
65	101	Speech Data Base at LL-TX-2	[23]
67	103	Datacomputer at CCA	[24]
69	105	CPYNET	
71	107	NETRJS (EBCDIC) at UCLA-CCN	[25]
73	111	NETRJS (ASCII) at UCLA-CCN	[25]
75	113	NETRJS (TTY) at UCLA-CCN	[25]
77	115	any private RJE server	
79	117	Finger	
81	121	Network BSYS	
95	137	SUPDUP	[33]
<hr/>			
		Experimental Functions	
229	345	Garlick's Debugger	
232-237	350-355	Authorized Mailer at BBN	
239	357	Graphics	[1, 26]
241	361	NCP Measurement	[27, 28]
243	363	Survey Measurement	[28, 29, 30]
245	365	LINK	[31]
247	367	TIPSRV	
249-255	371-377	RSEEXEC	[31, 32]

Assigned Network Numbers

ASSIGNED NETWORK NUMBERS

This list of network numbers is used in the internetwork protocols now underdevelopment, the field is 8 bits in size.

Assigned Network Numbers

Decimal	Octal	Network
0	0	Reserved
1	1	BBN Packet Radio Network
2	2	SF Bay Area Packet Radio Network (1)
3	3	BBN RCC Network
4	4	Atlantic Satellite Network
5	5	Washington D.C. Packet Radio Network
6	6	SF Bay Area Packet Radio Network (2)
7-9	7-11	Not assigned
10	12	ARPANET
11	13	University College London Network
12	14	CYCLADES
13	15	National Physical Laboratory
14	16	TELENET
15	17	British Post Office EPSS
16	20	DATAPAC
17	21	TRANSPAC
18	22	LCS Network
19	23	TYMNET
20-254	24-376	Unassigned
255	377	Reserved

Assigned Numbers

RFC 739, NIC 42423 (Nov. 11, 1977)

Assigned Internet Message Versions

ASSIGNED INTERNET MESSAGE VERSIONS

In the internetwork protocols there is a field to identify the version of the internetwork general protocol. This field is 4 bits in size.

Assigned Internet Message Versions

Decimal	Octal	Version	References
-----	-----	-----	-----
0	0	Old	[35]
1	1	Current	[36]
2-14	2-16	Unassigned	
15	17	Reserved	

Assigned Internet Message Formats

ASSIGNED INTERNET MESSAGE FORMATS

In the internetwork protocols there is a field to identify the format of the host level specific protocol. This field is 8 bits in size.

Assigned Internet Message Formats

Decimal	Octal	Format	References
0	0	Reserved	
1	1	raw internet	
2	2	TCP-3	[36]
3	3	DSP	[37,38]
2-254	2-376	Unassigned	
255	377	Reserved	

Assigned Numbers

RFC 739, NIC 42423 (Nov. 11, 1977)

Assigned Internet Message Types

ASSIGNED INTERNET MESSAGE TYPES

In the internetwork old protocol there is a field to identify the type of the message. This field is 4 bits in size.

Assigned Internet Message Types

Decimal	Octal	Type
0	0	Escape
1	1	TCP-2
2	2	Secure
3	3	Gateway
4	4	Measurement
5	5	DSP
6	6	UCL
7-12	7-14	Reserved
13	15	Pluribus
14	16	Telenet
15	17	Xnet

References

REFERENCES

- [1] Feinler, E. "ARPANET Protocol Handbook," NIC 7104, Defense Communications Agency, 1 April 1976.
- [2] BBN, "Specifications for the Interconnection of a Host and an IMP," Report 1822, Bolt Beranek and Newman, Cambridge, Massachusetts, January 1976.
- [3] McKenzie, A. "Host/Host Protocol for the ARPA Network," NIC 8246, January 1972.
- [4] Walden, D. "A System for Interrprocess Communication in a Resource Sharing Computer Network," RFC 62, NIC 4962, 3-Aug-70. Also published in Communications of the ACM volume 15, number 4, April 1972.
- [5] Bressler, B. "A Proposed Experiment with a Message Switching Protocol," RFC 333, NIC 9926, 15-May-72.
- [6] Postel, J. "Telnet Protocol," RFC 318, NIC 9348, 3-April-72.
- [7] McKenzie, A. "File Transfer Protocol," NIC 14333, RFC 454, 16-Feb-73.
- [8] Clements, R. "FTPSRV -- Extensions for Tenex Paged Files," RFC 683, NIC 32251, 3-April-75.
- [9] Harvey, B. "One More Try on the FTP," RFC 691, NIC 32700, 6-Jun-75.
- [10] Bressler, B. "Remote Job Entry Protocol," RFC 407, NIC 12112, 16-Oct-72.
- [11] Postel, J. "Echo Process," RFC 347, NIC 10426, 30-May-72.
- [12] Postel, J. "Discard Process," RFC 348, NIC 10427, 30-May-72.
- [13] Postel, J. "Character Generator Process," RFC 429, NIC 13281, 12-Dec-72.
- [14] Neigus, N. "File Transfer Protocol," NIC 17759 RFC 542 12-July-73.
- [15] Postel, J. "Revised FTP Reply Codes," NIC 30843 RFC 640 5-June-74.

Assigned Numbers

RFC 739, NIC 42423 (Nov. 11, 1977)

References

- [16] McKenzie, A. "Telnet Protocol Specifications," NIC 18639, August 1973.
- [17] McKenzie, A. "Telnet Option Specification," NIC 18640, August 1973.
- [18] White, J. "A High Level Framework for Network-Based Resource Sharing," RFC 707, NIC 34263, 14 January 1976. Also in NCC Proceedings, AFIPS, June 1976.
- [19] White, J. "Elements of a Distributed Programming System," RFC 708, NIC 34353, 28 January 1976.
- [20] COMPASS. "Semi-Annual Technical Report," CADD-7603-0411, Massachusetts Computer Associates, 4 March 1976. Also as, "National Software Works, Status Report No. 1," RADC-TR-76-276, Volume 1, September 1976. and COMPASS. "Second Semi-Annual Report," CADD-7608-1611, Massachusetts Computer Associates, 16 August 1976.
- [21] NSW Protocol Committee, "MSG: The Interprocess Communication Facility for the National Software Works," CADD-7612-2411, Massachusetts Computer Associates, BBN 3237, Bolt Beranek and Newman, Revised 24 December 1976.
- [22] Harrenstien, K. "Time Server," RFC 738, NIC 42218, 31-Oct-77.
- [23] Armenti, A., D. Hall, and A. Stone. "Lincoln Speech Data Facility," SUR Note 37, NIC 10917, 14 July 1972.
- [24] CCA, "Datacomputer Version 1 User Manual," Computer Corporation of America, August 1975.
- [25] Braden, R. "Interim NETRJS Specification," RFC 189, NIC 7133, 15-July-71.
- [26] Sproull, R, and E. Thomas. "A Networks Graphics Protocol," NIC 24308, 16-Aug-74.
- [27] Cerf, V., "NCP Statistics," RFC 388, NIC 11360, 23 August 1972.
- [28] Cerf, V., "Formation of a Network Measurement Group (NMG)," RFC 323, NIC 9630, 23 March 1972.

References

- [29] Bhushan, A., "A Report on the Survey Project," RFC 530, NIC 17375, 22 June 1973.
- [30] Cantor, D., "Storing Network Survey Data at the Datacomputer," RFC 565, NIC 18777, 28 August 1973.
- [31] Bressler, R., "Inter-Entity Communication -- An Experiment," RFC 441, NIC 13773, 19 January 1973.
- [32] Thomas, R. "A Resource Sharing Executive for the ARPANET," AFIPS Conference Proceedings, 42:155-163, NCC, 1973.
- [33] Crispin, M. "SUPDUP Protocol," RFC 734, NIC 41953, 7 October 1977.
- [34] Postel, J. "Official Initial Connection Protocol," NIC 7101, 11 June 1971.
- [35] Cerf, V. "Specification of Internet Transmission Control Program -- TCP (version 2)," March 1977.
- [36] Cerf, V. and J. Postel, "Specification of Internet Transmission Control Program -- TCP-3," November 1977.
- [37] Reed, D. "Protocols for the LCS Network," Local Network Note 3, Laboratory for Computer Science, MIT, 29 November 1976.
- [38] Clark, D. "Revision of DSP Specification," Local Network Note 9, Laboratory for Computer Science, MIT, 17 June 1977.



**NSW BIBLIOGRAPHY**  
**NIC 29613 (Dec. 13, 1977)**

J. Postel (USC-ISI)  
NIC 29613 (Dec., 13 1977)

**NSW BIBLIOGRAPHY**

Andrews, D., Boli, B. and Poggio, A.

An Introduction to the Frontend, Augmentation Research Center, SRI International, Menlo Park, Calif., Feb., 3 1977. (NIC 28743)

Andrews, D.

Command Sequence Processor Design Specification. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 28, 1977. (NIC 29046)

Andrews, D.

Frontend Communication Conventions. Augmentation Research Center, SRI International, Menlo Park, Calif., Mar. 10, 1976. (NIC 27719)

Andrews, D.

NSW Telnet-Frontend Specifications, Augmentation Research Center, SRI International, Menlo Park, Calif., Apr. 17, 1976. (NIC 27900).

Andrews, D.

System Configurations for Interactive Tools, Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 28, 1977 (NIC 29045).

Andrews, D., Boli, B., and Poggio, A.

A Guide to the Command Meta Language. Augmentation Research Center, SRI International, Menlo Park, Calif., Feb. 3, 1977. (NIC 28744)

Andrews, D.

Frontend Performance Analysis. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 21, 1977. (NIC 29025)

Balzer, R., Cheatham, T., Crocker, S. and Warshall, S.

Design of a National Software Works. Information Sciences Institute, Univ. Southern Calif., Marina Del Rey, Nov. 1973. (ISI-RR-73-16)

Bearisto, D. B.

A Description of the COMPASS Front End. Massachusetts Computer Associates, Inc., Wakefield, Mass., Nov. 19, 1976.

Preceding page blank

**NSW BIBLIOGRAPHY**  
**NIC 29613 (Dec. 13, 1977)**

Bearisto, D. B.

The Front End: User's Interface to the National Software Works. Massachusetts Computer Associates, Inc. Wakefield, Mass., Jan. 21, 1977. (CA-7701-2112)

Braden, R., and Ludlam, H.

An IP Server for NSW. Campus Computing Network, University of California, Los Angeles, CCN-TR-7, Apr. 1, 1976

Cashman, P. M., and Faneuf, R. A. and Muntz, C. A.

File Package: The File Handling Facility for the National Software Works (1st revision). Massachusetts Computer Associates, Inc., Wakefield, Mass., Dec. 27, 1976. (CADD-7612-2711)

COMPASS.

Second Semi-Annual Report. Massachusetts Computer Associates, Inc. CADD-7608-1611, August 16, 1976.

COMPASS.

Semi-Annual Technical Report. Massachusetts Computer Associates, Inc. CADD-7603-0411, March 4, 1976.

COMPASS.

Third Semi-Annual Technical Report for the National Software Works. Massachusetts Computer Associates, Inc. CADD-7702-2811, February 28, 1977.

Faneuf, R. A.

The National Software Works: Operational Issues in a Distributed Processing System. Massachusetts Computer Associates, Inc., Wakefield, Mass., Aug. 19, 1977. (CA-7708-1911)

Geller, D. P.

NSW Managers' Guide, Preliminary, CADD-7605-3112, Massachusetts Computer Associates, Wakefield, Mass., May 31, 1976.

Geller, D. P.

Interim NSW Managers Tools. Massachusetts Computer Associates, Inc. CA-7609-2912, Sept. 29, 1976.

Geller, D. P.

Interim NSW User System. Massachusetts Computer Associates, Inc., Wakefield, Mass., Sept. 29, 1976. (CA-7609-2911)

**NSW BIBLIOGRAPHY**  
**NIC 29613 (Dec. 13, 1977)**

Geller, D. P.

NSW Functional Test Plan/Procedures. Massachusetts Computer Associates, Inc., Wakefield, Mass., June 30, 1977. (CADD-7706-3011)

Geller, D. P.

NSW Manager's Guide (Preliminary). Massachusetts Computer Associates, Inc., Wakefield, Mass., May 31, 1976. (CADD-7605-3112)

Geller, D. P.

NSW User's Guide (Preliminary). Massachusetts Associates, Inc., Wakefield, Mass., May 31, 1976. (CADD-7605-3111)

Geller, D. P.

Preliminary NSW Manager's Guide. Massachusetts Associates, Inc., Wakefield, Mass., June 11, 1976. (CADD-7606-1112)

Geller, D. P.

Preliminary NSW User's Guide Volume II: NSW Language Reference Manual. Massachusetts Associates, Inc., Wakefield, Mass., Dec. 30, 1977. (CADD-7612-3011)

Geller, D. P.

Preliminary NSW User's Guide. Massachusetts Associates, Inc., Wakefield, Mass., June 11, 1976. (CADD-7606-1111)

Geller, D. P.

The National Software Works. Massachusetts Associates, Inc., Wakefield, Mass., Aug. 11, 1977. (CA-7708-1111)

Guerrieri, M., Schaffner, S. and Sluizer, S.

Works Manager Program Maintenance Manual. Massachusetts Computer Associates, Inc., Wakefield, Mass., Sept. 27, 1977. (CADD-7709-2711)

Guerrieri, M., Schaffner, S. and Sluizer, S.

Works Manager Subsystem Specification. Massachusetts Computer Associates, Inc., Sept. 27, 1977. (CADD-7709-2712)

Irby, C.

CLI Implementation Memo. Augmentation Research Center, SRI International, Menlo Park, Calif., June 3, 1976. (NIC 28184)

**NSW BIBLIOGRAPHY**  
**NIC 29613 (Dec. 13, 1977)**

Irby, C.

Control characters used in the current CLI. Augmentation Research Center, SRI International, Menlo Park, Calif., Nov. 7, 1975. (NIC 26891)

Irby, C.

NSW Frontend Packages and Procedures. Augmentation Research Center, SRI International, Menlo Park, Calif., July 17, 1975. (NIC 26151)

Irby, C.

The Command Meta Language System, Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 6, 1976. (NIC 27266)

Kremers, J.

Comparison of ELF and RSX-11 for Frontend Support, Augmentation Research Center, SRI International, Menlo Park, Calif., Feb. 10, 1976. (NIC 27569)

Kremers, J.

Input Output Station. Augmentation Research Center, SRI International, Menlo Park, Calif., Apr. 17, 1976. (NIC 27899)

Millstein, R. E.

The National Software Works. Massachusetts Computer Associates, Inc., Wakefield, Mass., July 30, 1976. (CA-7607-3011)

Millstein, R. E.

The National Software Works: A Distributed Processing System. Massachusetts Computer Associates, Inc., Wakefield, Mass., Aug. 9, 1977. (CA-7708-0911)

Millstein, R. E. and Schantz, R. E.

NSW Tool Builder's Guide (1st revision). Massachusetts Computer Associates, Inc., Wakefield, Mass. and Bolt, Beranek and Newman, Inc., Cambridge, Mass., Feb. 18, 1977. (CADD-7702-1811, or BBN 3308)

Millstein, R. E. and Schantz, R. E.

NSW Tool Builder's Guide (Preliminary). Massachusetts Computer Associates, Inc., Wakefield, Mass. and Bolt, Beranek and Newman, Inc., Cambridge, Mass., May 21, 1976. (CADD-7702-1811, or BBN 3308)

Millstein, R. E. and Shapiro, R. M.

Interim NSW Reliability Plan (1st revision). Massachusetts Computer Associates, Inc., Wakefield, Mass., Feb. 8, 1977. (CA-7701-2111)

NSW BIBLIOGRAPHY  
NIC 29613 (Dec. 13, 1977)

Millstein, R.E., and Shapiro, R.M.

Interim NSW Reliability Plan (2nd revision). Massachusetts Computer Associates, Inc., Wakefield, Mass., March 1, 1977. (CA-7701-2111)

Millstein, R.E., and Shapiro, R.M.

Interim NSW Reliability Plan. Massachusetts Computer Associates, Inc., Wakefield, Mass., Jan. 21, 1977. (CA-7701-2111)

Muntz, C.A., and Cashman, P.M.

File Package: The File Handling Facility for the National Software Works (Preliminary). Massachusetts Computer Associates, Inc., Wakefield, Mass., Feb. 20, 1976. (CADD-7602-2011)

NSW Protocol Committee.

MSG: The Interprocess Communication Facility for the National Software Works (1st revision). Massachusetts Computer Associates, Inc., Wakefield, Mass. and Bolt, Beranek and Newman, Inc., Cambridge, Mass., Dec. 24, 1976. (CADD-7612-2411, BBN 3483)

NSW Protocol Committee.

MSG: The Interprocess Communications Facility for the National Software Works (Preliminary). Massachusetts Computer Associates, Inc., Wakefield, Mass. and Bolt, Beranek and Newman, Inc., Cambridge, Mass., Jan. 23, 1976. (CADD-7601-2611, BBN 3237)

\* Ocken, S., and Poggio, A.

Scenario for Tool Initialization and Termination. Augmentation Research Center, SRI International, Menlo Park, Calif., July 21, 1976. (NIC 28409)

Poggio, A.

Error Reporting Standards. Augmentation Research Center, SRI International, Menlo Park, Calif., May 13, 1976. (NIC 28074)

Postel, J. and Weinberg, A.

Debugger Program Documentation. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 5, 1977. (NIC 28982)

Postel, J. and Weinberg, A.

Frontend Program Documentation. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 5, 1977. (NIC 28981)

**NSW BIBLIOGRAPHY**  
**NIC 29613 (Dec. 13, 1977)**

Postel, J. and Weinberg, A.

NLS Program Documentation. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 5, 1977. (NIC 28983)

Postel, J.

Description of Identification System Features., Augmentation Research Center, SRI International, Menlo Park, Calif., Oct. 13 1975. (NIC 26669)

Postel, J.

NSW Frontend Process Communication Conventions. Augmentation Research Center, SRI International, Menlo Park, Calif., May 14, 1976. (NIC 28075)

Postel, J.

NSW Tool-Frontend Communication. Augmentation Research Center, SRI International. NIC 28462, 5 August 1976.

Rome Air Development Center (RADC)

National Software Works, Status Rept No. 1., Vol. 1, Rome Air Development Center, Griffiss AFB, N. Y., Sept. 1976. (RADC-TR-76-276)

Rom, R.

Minimum Frontend Specification. Augmentation Research Center, SRI International, Menlo Park, Calif., Sept. 23, 1976. (NIC 28672)

Rom, R.

User Profile Users' Guide. Augmentation Research Center, SRI International, Menlo Park, Calif., Aug. 19, 1976. (NIC 28554)

Rom, R.

User Profile Data Structure Specification. Augmentation Research Center, SRI International, Menlo Park, Calif., Aug. 19, 1976. (NIC 28553)

Sattley, K.

NSW Prototype Management Tools - Maintenance Manual. Massachusetts Computer Associates, Inc., Wakefield, Mass., Nov. 4, 1976. (CADD-7611-0411)

Sattley, K.

NSW Prototype Management Tools - Subsystem Specification. Massachusetts Computer Associates, Inc., Wakefield, Mass., Oct. 7, 1976. (CADD-7610-0711)

NSW BIBLIOGRAPHY  
NIC 29613 (Dec. 13, 1977)

Schaffner, S.

BCPL Standards for NSW. Massachusetts Computer Associates, Inc., Wakefield, Mass., Aug. 23, 1977. (CADD-7708-2311)

Schaffner, S.

Works Manager Database Specification. Massachusetts Computer Associates, Inc., Wakefield, Mass., Sept. 27, 1977. (CADD-7709-2713)

Schantz, R. E. and Millstein, R. E.

The Foreman: Providing the Program Execution Environment for the National Software Works (1st revision). Bolt, Beranek and Newman, Inc., Cambridge, Mass. and Massachusetts Computer Associate, Inc., Wakefield, Mass., Jan. 1, 1977. (BBN 3442)

Schantz, R. E. and Millstein, R. E.

The Foreman: Providing the Program Execution Environment for the National Software Works (Preliminary). Bolt, Beranek and Newman, Inc., Cambridge, Mass. and Massachusetts Computer Associates, Inc., Wakefield, Mass., Mar. 31, 1976. (CADD-7604-0111, BBN 3266)

Shapiro, R. M. and Millstein R. E.

NSW Reliability Plan. Massachusetts Computer Associates, Inc., Wakefield, Mass., Jan. 14, 1977. (CA-7701-1411)

Shapiro, R. M. and Millstein, R. E.

NSW Reliability Plan (1st revision). Massachusetts Computer Associates, Inc., Wakefield, Mass., June 10, 1977. (CA-7701-1411)

Shapiro, R. M. and Millstein, R .E.

Plan for Hardening, Scaling and Optimizing the Works Manager. Massachusetts Computer Associates, Inc., Wakefield, Mass., January 15, 1976. (CADD 7601-1511)

Shapiro, R. M. and Millstein, R. E.

Reliability and Fault Recovery in Distributed Processing. Massachusetts Computer Associates, Inc., Wakefield, Mass., August 18, 1977. (CA-7708-1811)

Warshall, S.

NSW: Tools for Management Support (a final report). Massachusetts Computer Associates, Inc., Wakefield, Mass., July 1, 1976. (CADD-7607-0111)

Warshall, S.

NSW: Tools for Management Support (1st revision). Massachusetts Computer Associates, Inc., Wakefield, Mass., April 8, 1976. (CADD-7604-0811)

**NSW BIBLIOGRAPHY**  
**NIC 29613 (Dec. 13, 1977)**

Warshall, S.

NSW: Tools for Management Support. Massachusetts Computer Associates, Inc. Wakefield, Mass., February 2, 1976. (CADD-7602-0211)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

**BIBLIOGRAPHY**

Abramson, N.

The Aloha system. Tech. Rept. B72, Univ. of Hawaii, Honolulu, Jan. 1972. 16 pp. Also in: Abramson, N. and Kuo, F. F. Computer Communication Networks. Prentice-Hall, Englewood Cliffs, N. J., 1973. pp 501-17.

Abramson, N.

The ALOHA System, another alternative for computer communications. AFIPS Proc., Vol. 37, 281-5 (1970).

Abramson, N.

Digital broadcasting in Hawaii – the ALOHA system. EDUCOM Bull., Interuniversity Commun. Council, Vol. 9, No. 1, 9-13 (1974).

Abramson, N.

Packet switching with satellites. AFIPS Proc., Vol. 42, 695-702 (1973). Also: Tech. Rept. B73-2, Univ. of Hawaii, Honolulu, Mar. 1973. 28 pp. (AD-761544)

Abramson, N. and Kuo, F. F.

Computer Communication Networks. Prentice-Hall, Englewood Cliffs, N. J., 1973.

Akkoyunlu, E. A., Bernstein, A. J. and Schantz, R. E.

Interprocess communication facilities for network operating systems. Computer, Vol. 7, No. 6, 46-55 (June 1974).

Akkoyunlu, E. A., Ekanadham, K. and Huber, R. V.

Some constraints and tradeoffs in the design of network communications. IN: Proc. Fifth Symposium on Operating Systems Principles, Austin, Tex., Nov. 1975. pp 67-74.

Anderson, R. H., Harslem, E. F., Heafner, J. F., Cerf, V. G.,  
Madden, J., Metcalfe, R. M., Soshani, A., White, J. and Wood, D. C.

Data reconfiguration service – an implementation specification. RFC 166, RAND Corp., Santa Monica, Calif., May 25, 1971. 27 pp. (NIC-06780)

Anderson, R. H., Harslem, E. F., Heafner, J. F., Cerf, V. G.,  
Madden, J., Metcalfe, R. M., Soshani, A., White, J. and Wood, D. C.

Data reconfiguration service – compiler/interpreter. RFC 194, RAND Corp., Santa Monica, Calif., July 1971. 22 pp. (NIC-07139)

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Anderson, R. H., Harslem, E. F., Heafner, J. F., Cerf, V. G.,  
Madden, J., Metcalfe, R. M., Soshani, A., White, J. and Wood, D. C.

The data reconfiguration service, an experiment in adaptable process/process communication.  
IN: ACM/IEEE Second Symposium on Problems in the Optimization of Data Communication  
Systems, Oct. 20-22, 1971. pp 1-9. (IEEE CAT-71C59-C) (AD-735078)

Anderson, R. H., Harslem, E. F., Heafner, J. F., Cerf, V. G.,  
Madden, J., Metcalfe, R. M., Soshani, A., White, J. and Wood, D. C.

Status report on proposed data reconfiguration service. RFC 138, RAND Corp., Santa Monica,  
Calif., Apr. 28, 1971. 30 pp. (NIC-06715)

Anypan, N. S. and Kotob, S.

Analysis of flow control in AUTODIN II. IN: NTC'77 Conference Record, Vol. 3, IEEE, Inc.,  
New York, N. Y. (1977). pp 37:3,1-6 (IEEE # 77CH1292-2 CSCB)

Baran, P.

On distributed communications networks. IEEE Trans. Commun., Vol. 12, No. 1, 1-9 (Mar.  
1964).

Baran, P., Boehm, S. and Smith, P.

Determination of path-lengths in a distributed network. RAND Corp., Santa Monica, Calif.,  
Aug. 1964. 91 pp. (AD-444833)

Baran, P., Boehm, S. and Smith, P.

Digital simulation of hot-potato routing in a broadband distributed communication network.  
RAND Corp., Santa Monica, Calif., Aug. 1964. 49 pp. (AD-444834)

Baran, P., Boehm, S. and Smith, P.

Priority, precedence, and overload. RAND Corp., Santa Monica, Calif., Aug. 1964. 63 pp.  
(AD-444840)

Baran, P., Boehm, S. and Smith, P.

Tentative engineering specifications and preliminary design for a high-data-rate distributed  
network switching node. RAND Corp., Santa Monica, Calif., Aug. 1964. 85 pp.  
(AD-444832)

Barber, D. L. A.

Digital interfacing. IN: Paker, Y., Cain, G., and Morse, P., Proc. Minicomputer Interfacing,  
Miniconsult, London, England, 1973. pp 37-70.

Barber, D. L. A.

A specification for a European Informatics Network. Co-Operation Européenne dans le  
domaine de la Recherche Scientifique et Technique, Jan. 4, 1974.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Barber, D. L. A.

Easing the introduction of a packet switching service. Div. of Computer Science, National Physical Lab., Teddington, England, Mar. 1971. 20 pp

Barber, D. L. A.

Experience with the use of the B.S. interface in computer peripherals and communication systems. Div. of Computer Science, National Physical Lab., Teddington, England, Oct 1969. 15 pp.

Barber, D. L. A.

The European Computer Network project. Proc. ICCC, Washington, D. C., 192-200 (Oct. 1972).

Barber, D. L. A.

The European Informatics network: Achievements and prospects. Proc. ICCC, Toronto, Canada (1976).

Barber, D. L. A.

Progress with the European Informatics Network. Proc. ICCC, Stockholm, Sweden (1974). pp 215-20.

Bauwens, E. and Magnee, F.

A virtual terminal prototcol for a Belgian University Network. INWG Protocol Note # 71, International Network Working Group. 64 pp.

Bauwens, E. and Magnee, F.

The virtual terminal approach in the Belgian University Network. INWG Protocol Note # 83, International Network Working Group. June 1977. 196 pp.

Beeforth, T. H., et. al

Proposed organization for a packet-switched data-communication network. Proc. IEE, Vol. 119, No. 12, 1677-82 (Dec. 1972).

Beere, M. and Sullivan, N.

TYMNET--A serendipitous evolution. IN: Proc. Second ACM/IEEE Symposium on Problems in the Optimization of Data Communications, Palo Alto, Calif., Oct 1971.

Belloni, A., Bozetti, M. and LeMoli, G.

Routing and internetworking. Alto Frequenza, Vol. 44, No. 4, 194-210 (Apr. 1975). (INWG Protocol Note 10)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Belsnes, D.

Flow control in packet switching networks. DSL Tech. Note #50, Digital Systems Lab., Stanford Univ., Calif., Oct. 1974. (INWG Protocol Note #63)

Belsnes, D.

Note on single message communication. Digital Systems Lab., Stanford Univ., Calif., Sept. 1974. (INWG Protocol Note #3)

Benice, R. J. and Frey, A. H.

An analysis of retransmission systems. IEEE Trans. Commun. Technol., Vol. 12, No. 4, 135-145 (Dec. 1964).

Benice, R. J. and Frey, A. H.

Comparison of error control techniques. IEEE Trans. Commun. Technol., Vol. 12, No. 4, 146-154 (Dec. 1964).

Benoit, J. W. and Wood, D. C.

Network front end study. Tech. Rept. MRT-5213, MITRE Corp., Washington, D.C., Aug. 1974. (Limited Distribution)

Benoit, J. W.

Evolution of network user services--the network resource manager. IN: Proc. 1974 NBS-IEEE Symp. Computer Networks: Trends and Applications, Gaithersburg, Md., May 1974. pp 21-4.

Benoit, J. W. and Perez, E.

Design specifications for PWIN non-functional network control software. MITRE Corp., Washington, D.C., June 30, 1972. 245 pp.

Berkheiser, E. W., Baker, P. C., Capaldo, R.

Security architecture for autodin II. IN: NTC'77 Conference Record, Vol. 3, IEEE, Inc., New York, N. Y. (1977). p 37:4,1-6  
(IEEE 77CH1292-2 CSCB)

Bhushan, A. K.

Another look at data and file transfer protocols. RFC 310, Proj. MAC, Massachusetts Inst. Technol., Cambridge, Mass., Apr. 3, 1972. 7 pp. (NIC-09261)

Bhushan, A. K., Kanodia, R., Metcalfe, R. M. and Postel, J. B.

Comments on byte size for connections. RFC 176, Massachusetts Inst. Technol., Cambridge, Mass., June 14, 1971. 5 pp. (NIC-07100)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Bhushan, A. K., Kanodia, R., Metcalfe, R. M. and Postel, J. B.

The data transfer protocol. RFC 264, Dynamic Modeling/Computer Graphics Systems, Massachusetts Inst. Technol., Cambridge, Mass., Nov. 15, 1971. 8 pp. (NIC-07812)

Binder, R.

A dynamic packet switching system for satellite broadcast channels. ALOHA System Tech. Rept. B74-5, Univ. of Hawaii, Honolulu, Aug. 1974.

Binder, R.

Another ALOHA satellite protocol. ALOHA System, Univ. of Hawaii, Honolulu, Dec. 27, 1972. (NIC-13147)

Binder, R.

Multiplexing in the ALOHA system: Menehune-Keiki design considerations. ALOHA System Tech. Rept. B69-3, Univ. of Hawaii, Honolulu, Nov. 1969. 60 pp. (AD-702807)

Binder, R., Abramson, N., Kuo, F. F., Okinaka, A. and Wax, D.

ALOHA packet broadcasting - a retrospect. AFIPS Proc., Vol. 44, 203-15 (1975).

Binder, R., Lai, W. S., Wilson, M.

The ALOHANET Menehune - version II. ALOHA System Tech. Rept. B74-6, Univ. of Hawaii, Honolulu, Sept. 1974.

Binder, R., Rettberg, R. and Walden, D.

The Atlantic satellite packet broadcast and gateway experiments. BBN Rept. 3056, Bolt Beranek and Newman, Cambridge, Mass., Apr. 1975.

Bochmann, G. V.

Finite state description of communication protocols, Publ. 236, Dep. d'Informatique, Univ. de Montreal, July 1976.

Bochmann, G. V.

Communication protocols and error recovery procedures, Proc. ACM Interprocess Commun. Workshop, Mar. 1975.

Bochmann, G. V.

Logical verification and implementation of protocols. Publ. 190, Department d'Informatique, Universite de Montreal, Mar. 1975. (Also in Proc. Fourth Data Communications Symposium, Quebec City, Canada, Oct. 1975., pp 7-15 to 7-20. IEEE 75 CH1001-7 DATA)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Bochmann, G. V.

Proving the correctness of communication protocols. Tech. Rept., Department d'Informatique, Universite de Montreal, Nov. 1974.

Bochmann, G. V. and Chung, R. J.

A Formalized Specification of HDLC Classes of Procedures. IN: NTC'77 Conference Record, Vol. 1, IEEE, Inc., New York, N. Y., 1977. pp 3A:2,1-4 (IEEE # 77CH1292-2 CSCB)

Bochmann, G. V. and Gecsei, J.

A unified method for the specification and verification of protocols. Proc. IFIP Congress, Vol. 7, 229-234 (1977).

Boehm, B. W. and Mobley, R. L.

Adaptive routing techniques for distributed communications systems. IEEE Trans. Commun. Technol., Vol. 17, No. 3, 340-9 (June 1969). (Also AD-630271)

Bouknight, W. J.

The IMP interface manual. NTS Rept. 2, Network Tech. Systems Project, Univ. of Illinois, Urbana, July 1973.

Bouknight, W. J., Denenberg, S.

ANTS--A new approach to accessing the ARPA network. CAC Rept. 47, Center for Advanced Computation, Univ. of Illinois, Urbana, July 1972.

Bowdon, E. K. and Barr, W. J.

Cost effective priority assignment in network computers. Proc. Fall Joint Computer Conf., 1972, AFIPS Press, pp 755-63.

Braden, R. T.

NETRJT - Remote job service protocols for TIPS. RFC 283, Campus Computing Network, Univ. of California, Los Angeles, Dec. 20, 1971. 9 pp. (NIC-08165)

Braden, R. T.

A server host system on the ARPANET. pp 4:1-9 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Bressler, R. D. and Thomas, R. H.

FTP server-server interaction - II. RFC 478, Bolt Beranek and Newman, Cambridge, Mass., Mar. 26, 1973. (NIC-14947)

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Bressler, R. D., Murphy, D. and Walden, D. C.

A proposed experiment with a message switching protocol. RFC 333, Dynamic Modeling/Computer Graphics Systems, Mass. Inst. Tech., Cambridge, May 15, 1972. 42 pp. (NIC-09926)

Bright, R. D.

EPSS specification. INWG Protocol Note 1, Oct. 1974.

Bright, R. D.

Experimental packet switch project of the UK Post Office. IN: Grimsdale, R. L. and Kuo, F. F., Computer Communication Networks, Noordhoff International Publ., Leyden, Netherlands, 1975. pp 435-44.

Buckley, J. E.

Packet switching. Computer Design, Vol. 13, No. 4, 10-14 (Apr. 1974).

Burchfiel, J. D. and Tomlinson, R.

Proposed change to host-host protocol: resynchronization of connection status. RFC 467, Bolt Beranek and Newman, Cambridge, Mass., Feb 20, 1973. (NIC-14741)

Burchfiel, J. D., Tomlinson, R. and Beeler, M.

Functions and structure of a packet radio station. AFIPS Proc., Vol. 44, 245-51, (May 1975).

Butterfield, S., Rettberg, R. D. and Walden, D. C.

The satellite IMP for the ARPA network. IN: Seventh Hawaii International Conf. on System Sciences, Computer Nets Supplement, Jan. 1974. pp 70-3.

Carlstedt, J.

A message exchange for computer programs and terminals. RAND Corp., Santa Monica, Calif., May 1972. 70 pp. (AD-748986)

Carr, S., Crocker, S. D. and Cerf, V. G.

Host-Host communication protocol in the ARPA network. AFIPS Proc., Vol. 36, 589-98 (1970).

Casey, R. G.

Allocation of copies of a file in an information network. AFIPS Proc., Vol. 40 (1972).

Cashin, P. M.

Datapac Network protocols. IN: Proc. ICCC, Toronto, Canada, Aug. 1976. pp 150-55.

## BIBLIOGRAPHY

### ARPA/NET PROTOCOL HANDBOOK, NIC 7104

Cashin, P. M.

Datapac standard network access protocol. Computer Comm. Group, Trans-Canada Telephone System, 1974.

Cerf, V. G.

An assessment of ARPANET protocols. RFC 635, Digital Systems Lab., Stanford Univ., Calif., Apr. 22, 1974. (NIC 30489)

Cerf, V. G.

ARPA internetwork protocols project status rept. Tech. Note 68, Digital Systems Lab., Stanford Univ., Calif., Nov. 1975.

Cerf, V. G.

Host and process level protocols for internetwork communication, INWG Protocol Note 39, Digital Systems Lab, Stanford Univ., Calif., Sept. 1973.

Cerf, V. G. and Kahn, R. E.

A protocol for packet network intercommunication. IEEE Trans. Commun., Vol. COM-22, No. 5, 637-48 (May 1974).

Cerf, V. G. and Naylor, W. E.

Storage considerations in store-and-forward message switching. IN: 1972 WESCON Tech. Papers, Vol. 16, Session 7, Computer Networks, 1972, pp 7:3,1-8.

Cerf, V. G. and Sunshine, C.

Protocols and gateways for the interconnection of packet switching networks. ALOHA System Tech. Rept. CN74-22, Univ. of Hawaii, Honolulu, Jan 1974. (Also in Proc. Subconference Computer Networks, HICSS-7, Honolulu, Hawaii, Jan 1974.)

Cerf, V. G., Cowan, D. D., Mullin, R. C. and Stanton, R. G.

Topological design considerations in computer communication networks. IN: Grimsdale, R. L. and Kuo, F. F., Computer Communications Networks, NATO Advanced Studies Institute Series, E-4, Noordhoff Int., Leyden, Netherlands, 1975. pp 101-12.

Cerf, V. G., Dalal Y. and Sunshine, C.

Specification of internet transmission control program. INWG Protocol Note 72, Digital Systems Lab., Stanford Univ., Calif., revised Dec. 1974.

Cerf, V. G., McKenzie, A., Scantlebury, R. and Zimmerman, H.

Proposal for an internetwork end to end protocol. INWG Protocol Note 96, Digital Systems Lab., Stanford Univ., Calif., Sept. 1975.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Cerf, Vinton G., Harslem, Eric F., Heafner, John F.,  
Metcalfe, Robert M. and White, James E.

An experimental service for adaptable data reconfiguration. IEEE Trans. Commun., Vol. COM-20, No. 3, 557-64 (June 1972).

Chandler, A. S.

Network independent high level protocols. IN: Proc. EUROCOMP, Brunel Univ., Sept. 1975. pp 583-601.

Chedwick, H. D.

Maximum data throughput over digital transmission systems. IN: NTC'77 Conf. Record, Vol. 1, IEEE, Inc., New York, N. Y., 1977. pp 3B:2,1-4 (IEEE # 77CH1292-2 CSCB)

Chou, W. W.

Computer communication networks--The parts make up the whole. AFIPS Proc., Vol. 44, 119-27 (1975).

Chou, W. W. and Frank, H.

Routing strategies for computer network design. IN: International Symp. Comp. Commun. Networks and Teletraffic, Apr. 1972. pp 301-9.

Chu, W. W.

Optimal file allocation in a multiple computer system. IEEE Trans. Computers, Vol. C-18, 885-9 (Oct. 1969).

Chu, W. W. and Ohlmacher, G.

Avoiding deadlock in distributed data bases. Proc. ACM National Symposium, Vol. 1, 156-60 (Mar 1974).

Cohen, D.

Issues in transnet packetized voice communiation. pp 6:10-13 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Colon, F. C., Glorioso, R. M. and Li, D.

Analysis of a multiple processor system: A case study of DFMP. IN: IEEE 1976 Compcon Fall, Sep. 1976. pp 255-261.

Cosell, B. P., Johnson, P. R., Malman, J. H., Schantz, R. E.,  
Sussman J., Thomas, R. H. and Walden, D. C.

An operational system for computer resource sharing. IN: Proc. Fifth Symposium on Operating Systems Principles, ACM/SIGOPS, Austin, Tex., Nov. 1975. pp 25-80.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Cotton, I. W. and Folts, H. C.

International standards for data communications: A status report. pp 4:26-36 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Crocker, S. D.

Host-Host protocol document number 1. Network Working Group RFC???, Univ. of California, Los Angeles, Calif., Aug. 3, 1970. 17 pp (NIC-05143)

Crocker, S. D., Heafner, J. F., Metcalfe, R. M. and Postel, J. B.

Function-oriented protocols for the ARPA computer network. AFIPS Proc., Vol. 40, 271-9 (1972).

Crowther, W. R., Heart, F. E., McKenzie, A. A., McQuillan, J. M.  
and Walden, D. C.

Issues in packet-switching network design. AFIPS Proc., Vol. 44, 161-75 (May 1975).

Crowther, W. R., Rettberg, R. D., Walden, D. C., Ornstein, S. M.  
and Heart, F. E.

A system for broadcast communication: reservation ALOHA. IN: Hawaii International Conf. System Sciences, Univ. of Hawaii, Honolulu, Nov. 21, 1972. (NIC-12744)

Dalal, Y. K.

Establishing a connection. INWG Protocol Note 14, Digital Systems Lab., Stanford Univ., Calif., Mar. 1975.

Danthine, A.

Modeling and verification of end-to-end protocols. IN: NTC'77 Conference Record, Vol. 1, IEEE, Inc., New York, N. Y., 1977. pp 3A:3,1-7 (IEEE 77CH1292-2 CSCB)

Danthine, A.

Petri nets for protocol modeling and verification. INWG Protocol Note 84, Univ. de Liege, Belgium, Sept. 1977. 88 pp.

Danthine, A. and Bremer, J.

Communication protocols in a network environment. ACM Operating Systems Rev., Vol. 9, No. 3, (July 1975).

Danthine, A. and Bremer, J.

An axiomatic description of the transport protocol of Cyclades, Professional Conf. on Computer Networks and Teleprocessing, TH Aachen, Germany, Mar. 1976.

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Danthine, A. and Bremer, J.

Definition, representation, et simulation de protocoles dan un contexte reseaux. Journal Intern. Mini-ordinateurs et Transm. de Donnees, AIM, Liege, Jan. 1975.

Danthine, A. and Bremer, J.

Modeling and verification of end-to-end protocols. INWG Protocol Note 74, Univ. de Liege, Belgium, Apr. 1977. 156 pp.

Danthine, A. and Eschenauer, L.

Influence on the node behavior of the node-to-node protocol. IN: Proc. Fourth Data Communications Symposium, Quebec City, Canada, Oct. 1975, pp 7:1-8. (IEEE 75 CH1001-7 DATA) (INWG Protocol Note 22, Mar 1975).

Davidson, J.

An echoing strategy for satellite links. RFC 357, SRI International, Menlo Park, Calif., June 1972. (NIC-10599)

Davidson, J., Hathway, W., Postel, J., Mimno, N.,  
Thomas, R. and Walden, D.

The ARPANET Telnet protocol: Its purposes, principles, implementation, and impact on host operating system design. pp 4:10-18IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Davies, D. W.

Communication networks for computers. John Wiley, 1974.

Davies, D. W.

The control of congestion in packet switching networks. IEEE Trans. Commun., COM-20, No. 3, 546-50 (June 1972).

Davies, D. W.

Packet switching, message switching and future data communication networks. IFIP Congress 74, Vol. 1, 147-50 (Dec. 1974).

Day, J. D.

A Note on some unresolved issues for an end-to-end protocol. INWG Protocol Note 98, Center for Advanced Computation, Univ. of Illinois, Urbana, Aug. 1975.

Day, J. D.

Resilient protocols for computer networks. CAC Doc. 162 , Center for Advanced Computation, Univ. of Illinois, Urbana, May 1975. pp 25-70.

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Deparis, M., Duenki, A., Gien, M., Laws, J., Le Moli, G. and Weaving, K.

The implementation of an end-to-end protocol by EIN centres: A survey and comparison.  
Proc. Third International Conf. Computer Comm., Trans-Canada Telephone System, Aug. 1975.  
pp 351-60.

Despres, R. F.

A packet switching network with graceful saturated operation. IN: Proc. First International  
Conf. Comp. Commun., Oct. 1972, pp 345-51.

Dijkstra, E. W.

Self-stabilizing systems in spite of distributed control. Commun. ACM, Vol. 17, 643-4 (Nov.  
1974).

Donnan, R. A. and Kersey, J. R.

Synchronous data link control: A perspective. IBM Systems Journal, Vol. 13, No. 2, 140-62  
(1974).

EPSS Liaison Group

A basic file transfer protocol. INWG Protocol Note 93, June 1975.

EPSS Liaison Group

An interactive terminal protocol. INWG Protocol Note 94, June 1975.

Erskine, S. B.

Access to packet switching networks. IN: IEEE 1976 Compcon Spring, Feb 24, 1976. pp  
66-72.

Ewin, J. C. and Giloth, P. K.

No. 1 ESS ADF: system organization and objectives. Bell System Tech. J., 2733-52 (Dec  
1970).

Farber, D. J. and Larson, K.

The structure of a distributed computer system--communications. IN: Proc. Symp. Comp.  
Commun. Network and Teletraffic, MRI Symp. Proc., Vol. 22, Brooklyn Polytechnic Press,  
1972. pp 539-45.

Farber, D. J. and Larson, K.

The structure of a distributed computer system--software. IN: Proc. Symp. Comp. Commun.  
Network and Teletraffic, MRI Symp. Proc., Vol. 22, Brooklyn Polytechnic Press, 1972.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Farber, D. J. and Larson, K.

The structure of a distributed computer system--The distributed file system. IN: Proc. ICC - Impacts and Implications, Oct. 1972. pp 364-70.

Farber, D. J. and Vittal, J. J.

Extendability considerations in the design of the distributed computer system (DCS). IN: Proc. National Telecommunications Conf., Atlanta, Georgia, Nov. 1973.

Farber, D. J. and Larson, K. C.

The system architecture of the distributed computer system--the communications systems. In: Proc. Symp. Comp. Commun. Network and Teletraffic, Brooklyn Polytechnic Press, N. Y., 1972. pp 21-7.

Fayolle, G., Gelenbe, E., Labetoule, J. and Bastin, D.

Stability problem of broadcast packet switching networks. Acta informatica, Vol. 4, No. 1, 49-53 (1974).

Feinler, E. J. and Postel, J. B., eds.

ARPA network current network protocols. NIC-7104-REV-1, Augmentation Research Center, SRI International, Menlo Park, Calif., Dec. 1, 1974. 513 pp. (AD-A0038901)

Field, J. A. and Kalra, S. N.

Interfacing to packet switched networks. Proc. Third International Conf. Computer Comm., Trans-Canada Telephone System, Aug. 1975. pp 367-72.

Floyd, R. W.

Algorithm 97, shortest path. Commun. ACM, Vol. 5, No. 6, 345 (June 1962).

Forgie, J. W.

Speech transmission in packet-switched store-and-forward networks. AFIPS Proc., Vol. 44, 137-42 (1975).

Forgie, J. W. and McElwain, C. K.

ARPANET delay measurements. NSC Note 70, Lincoln Lab., Mass. Inst. Tech., Cambridge, July 1975.

Fralick, S. C. and Garrett, J. C.

Technological considerations for packet radio networks. AFIPS Proc., Vol. 44, 233-43 (1975).

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Frank, H.

Computer network design. IN: INTERCON 71, IEEE International Convention Digest 1971. pp 220-1.

Frank, H.

Optimal design of computer networks. IN: Randall, R., Courant Computer Science Symposium 3, 1970, Computer Networks, pp 167-183.

Frank, H.

Providing reliable networks with unreliable components. IN: DATACOMM 73: Data Networks Analysis and Design. Third Data Commun. Symp., 1973. pp 161-4.

Frank, H. and Chou, W.

Network properties of the ARPA computer network. Networks, Vol. 4, No. 3, 213-39 (1974)

Frank, H. and Chou, W.

Routing in computer networks. Networks, Vol. 1, No. 3, 99-112 (1971).

Frank, H. and Chou, W.

Topological optimization of computer networks. Proc. IEEE, Vol. 60, No. 11, 1385-96 (Nov 1972).

Frank, H. and Frisch, I. T.

Analysis and design of survivable networks. IEEE Trans. Commun. Technol., Vol. COM 18, 501-19 (Oct 1971).

Frank, H. and Frisch, I. T.

Design of large-scale networks. Proc. IEEE, Vol. 60, No. 1, 6-11 (Jan 1972).

Frank, H. and Frisch, I. T.

Design problems for computer networks. IN: Fifth Hawaii International Conf. System Sciences, Jan. 1972, pp 335-7.

Frank, H. and Frisch, I. T.

Planning computer - communication networks: the ARPA computer network. In: Abramson, N. and Kuo, F. F., Computer Communication Networks. Prentice-Hall, Englewood Cliffs, N. J., 1973. pp 17-28.

Frank, H. and Van Slyke, R. M.

Reliability considerations in the growth of computer communication networks. IN: National Telecommun. Conf., Vol. 2, Paper 22-D, 1973. 5 pp.

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Frank, H., Frisch, I. T. and Chou, W.

Topological considerations in the design of the ARPA computer network. AFIPS Proc., Vol. 36, 581-7 (1970). (NIC-04567)

Frank, H., Gitman, I. and Van Slyke, R.

Packet radio system--network considerations. AFIPS Proc., Vol. 44, 217-32 (1975).

Frank, H., Kleinrock, L. and Kahn, R. E.

Computer communication network design - experience with theory and practice. AFIPS Proc., Vol. 40, 255-70 (1972). (NIC-10273)

Frank, H., Van Slyke, R. and Gitman, I.

Packet radio network design - systems considerations. AFIPS Proc., Vol. 44, (1975).

Fratta, L., Gerla, M. and Kleinrock, L.

The flow deviation method: an approach to store-and-forward communication newtork design. Networks, Vol. 3, No. 2, 121-7 (1973).

Fredericksen, D. and Ryniker, R. W.

A computer network interface for OS/MVT. Watson Research Center, IBM, Yorktown Heights, N. Y., Apr. 5, 1971. 13 pp.

Fuchel, K. and Heller, S.

Two dissimilar networks--is marriage possible? IN: Proc. 1975 Symp. Computer Networks: Trends and Applications, 1975. pp 19-24.

Fultz, G. L.

Adaptive routing techniques for message switching computer-communication networks. UCLS-ENG-7252, Computer Science Department, School of Engineering and Applied Science, Univ. of Calif., Los Angeles, July 1972. 425 pp.

Fultz, G. L. and Kleinrock, L.

Adaptive routing techniques for store-and-forward computer communication networks. IN: International Conf. Commun., June 1971. pp 39:1-8.

Garlick, L., Rom, R., and Postel, J. B.

Reliable host-to-host protocols: Problems and techniques. IN: Fifth Data Commun. Symp., Snowbird, Utah, Sept. 1977.

Garlick, L. L., Rom, R., Postel, J. B.

Issues in reliable host-to-host protocols. IN: Proc. Second Berkeley Workshop on Distribued

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Data Management and Computer Networks, May 25-7, 1977, Lawrence Berkeley Lab., Univ. of Calif., Berkeley, 1977. pp 89-113.

Gaver, D. P. and Lewis, P. A. W.

Probability models for buffer storage allocation problems. J. ACM, Vol. 18, No. 2, 186-98, (Apr. 1971).

Georganas, N. D., Chatterjee, A. and Verma, P. K.

Local congestion control in computer-communication networks with random routing. pp 5:19-24 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Gerla, M.

Deterministic and adaptive routing policies in packet-switched computer networks. IN: Proc. Third ACM Data Commun. Symp., Nov 1973. pp 23-8.

Gerla, M.

The design of store-and-forward (S/F) networks for computer communications. Rept. ENG-7319, Computer Systems Modeling and Analysis Group, Computer Science Dept., Univ. of Calif., Los Angeles, Jan. 1973.

Gerla, M. and Chou, W.

Flow control strategies in packet switched computer networks. IN: National Telecommun. Conf. Record, IEEE, Inc., New York, N. Y., 1974. pp 1032-7.

Gerla, M. and Kleinrock, L.

Closed loop stability controls for S-ALOHA satellite communications. pp 2:10-19 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Gien, M.

Proposal for a standard file transfer protocol (FTP). INWG Protocol Note 73, May 1977. 188 pp.

Gien, M., Laws, J. and Scantlebury, R.

Interconnection of packet-switched networks: Theory and practice. IN: EUROCOMP, Brunel Univ., Sep. 1975. pp. 241-60.

Gilbert, P. and Chandler, W. J.

Interference between communicating parallel processes. Commun. ACM, Vol. 15, No. 6, 427-37 (June 1972).

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Gouda, M. G. and Manning, E. G.

Protocol machines: A concise formal model and its automatic implementation. IN: Proc. Third International Conf. Computer Comm., Trans-Canada Telephone System, Aug. 1976. pp 346-50.

Gouda, M. G. and Manning, E. G.

Toward modular hierarchical structures for protocols in computer networks. IN: IEEE 1976 Compcon Fall, Sept. 1976. pp 246-50.

Graham, R. L. and Pollack, H. O.

On the addressing problem for loop switching. Bell System Tech. J., Vol. 50, No. 8, 2495-519 (Oct. 1971).

Grimsdale, R. L. and Kuo, F. F. (Eds.)

Computer communication networks. Noordhoff International Publ., Leyden, Netherlands, 1975. 490 pp.

Guttag, J. V., Horowitz, E. and Musser, D. R.

The design of data type specifications. Information Sciences Institute, Univ. of Southern California, Nov. 1976.

Guttag, J. V., Horowitz, E. and Musser, D. R.

Protocol analysis of man-computer languages: design and preliminary findings. Information Sciences Institute, Univ. of Southern California, July 1976.

Harangozo, J.

An approach to describing a data link level protocol with a formal language. pp 4:37-49 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Harslem E. F., Heafner, J. F. and Wisniewski, T. D.

Data reconfiguration service compiler: communications among heterogenous computer centers using remote resource sharing. RAND Corp., Santa Monica, Calif., Apr. 1972. 127 pp. (AD-750283)

Harslem, E. F. and Heafner, J. F.

The data reconfiguration service - an experiment in adaptable, process/process communication. RAND Corp., Santa Monica, Calif., Nov. 1971. 29 pp. (AD-745751)

Hassing, T. E., Hampton, R. M., Bailey, G. W. and Gardella, R. S.

A loop network for general purpose data communications in a heterogeneous world. IN: Proc. Third Data Commun. Symp., Nov. 1973.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Heafner, J. F.

Protocol analysis of man-computer languages: design and preliminary findings. ISI/RR75-34, Information Sciences Inst., Univ. of Southern Calif., Marina del Rey, June 1975.

Heart, F. E.

The ARPA network. IN: Gimsdale, R. L. and Kuo, F. F., Computer Communication Networks. Leyden, the Netherlands, Noordhoff International Publishing, 1975. pp 19-33. (NIC-04565)

Heart, F. E., Kahn, R. E., Ornstein, S. M., Crowther, W. R. and Walden, D. C.

The interface message processor for the ARPA computer network. AFIPS Proc., Vol. 36, 551-567 (1970).

Heart, F. E., Ornstein, S. M., Crowther, W. R. and Barker, W. B.

A new minicomputer/multiprocessor for the ARPA network. AFIPS Proc., Vol. 42, 592-637 (June 1973).

Henderson, D. A. and Myer, T. H.

Issues in message technology. pp 6:1-9 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Higginson, P. L. and Hinchley, A. J.

The problems of linking several networks with a gateway computer. IN: Proc. European Comp. Conf. Commun. Networks, Online Conferences Ltds, Uxbridge, England, Sept. 1975. pp 25-34.

Holt, R. C.

Some deadlock properties of computer systems. Computing Surveys, Vol. 4, No. 4, 179-96 (Dec. 1972).

Hopwood, M. D., Loomis, D. C. and Rowe, L. A.

The design of a distributed computing system. Tech. Rept. 25, Dept. of Information and Computer Science, Univ. of Calif., Irvine, June 1973.

Huynh, D., Kobayashi, H. and Kuo, F. F.

Optimal design of mixed media packet switching networks: Routing and capacity assignment. IEEE Trans. Commun., Vol. 25, No. 1, 158-68 (1977).

Huynh, P., Kobayashi, H. and Kuo, F. F.

Design issues for mixed media packet switching networks. AFIPS Proc., Vol. 45, 541-9 (1976).

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

INWG Protocol Committee

Basic message format for internetwork communication. INWG Protocol Note 83, May 1975.

INWG Protocol Committee

Experiment in inter-networking. Basic message format. INWG Protocol Note 1, Nov. 1974.

INWG Protocol Committee

Timing parameters of packet switched data networks--user implications. INWG Protocol Note #82, May 1975.

Jackson, P. E., and Stubbs, C. D.

A study of multi-access computer communications. AFIPS Proc., Vol. 34, 491-504 (1969).

Jacobs, I., Lee, L., Viterbi, A., Binder, R., Bressler, R.,  
Hsu, N. and Weissler, R.

CPODA - a demand assignmnt protocol for SATNET. pp 2:5-9 IN: Fifth Data Commun.  
Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Kahn, R. E.

Resource-sharing computer communications networks. Proc. IEEE , Vol. 60, No. 11,  
1397-1407 (Nov 1972).

Kahn, R. E.

The ARPA network - packet switching technology. IN: IEEE Commun. Systems and Tech.  
Conf., 1974, p. 134.

Kahn, R. E.

The organization of computer resources into a packet radio network. AFIPS Proc., Vol. 44,  
177-186 (1975).

Kahn, R. E. and Crowther, W. R.

Flow control in a resource-sharing computer network. IEEE Trans. Commun., Vol. COM-20,  
No. 3, 539-546 (June 1972).

Kahn, R. E. and Crowther, W. R.

TIM/IEEE Second Symposium on Problems in the Optimization of Data Communications  
Systems, IEEE, Inc., New York, N. Y., 1971, pp 108-116. (NIC 11750)

Kalin, R. B.

A simplified NCP protocol. RFC 060, MIT Lincoln Lab., Lexington, Mass. ,July 13, 1970. 10  
pp. (NIC-04762)

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Karp, D. P. and Seroussi, S. F.

A communications interface for computer networks. IEEE Trans. Commun., COM Vol. 22, No. 3, Part 2, 550-6 (June 1972). (Also in Watson Research Center, IBM, Yorktown Heights, N. Y., June 24, 1971. 15 pp.)

Kent, S. T.

Encryption-based protection for interactive user/computer communication. IN: pp 5:7-13 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Kimbleton, S. R. and Schneider, G. M.

Computer communication networks: Approaches, objectives and performance considerations. Computing Surveys, Vol. 7, No. 3, 128-73 (Sept. 1967).

Kimbleton, S. R. and Mandell, R. L.

A perspective on network operating systems. AFIPS Proc., Vol. 45, 551-9 (1976).

Kleinrock, L.

Resource allocation computer systems and computer communication networks. Proc. IFIP Congress 74, Vol. 4, 11-18 (1974).

Kleinrock, L. and Lam, S. S.

On stability of packet switching in a random multi-access broadcast channel. IN: Seventh Hawaii International Conf. System Sciences, Jan 1974, Computer Nets Supplement. pp 74-7.

Kleinrock, L. and Lam, S. S.

Packet switching in a slotted satellite channel. AFIPS Proc., Vol. 42, 703-10 (1973).

Kleinrock, L. and Opderbeck, H.

Throughput in the ARPANET--protocols and measurement. IN: Fourth Data Commun. Symp., Quebec, Canada, Oct. 1975. (IEEE Catalog No. 75 CH 1001-7 DATA)

Kleinrock, L. and Tobagi, F.

Packet switching in radio channels: Part I--carrier sense multiple access modes and their throughput delay characteristics, IEEE Trans. Commun., Vol. 23, 1400-16 (Dec. 1975).

Kleinrock, L. and Tobagi, F.

Random access techniques for data transmission over packet-switched radio channels. AFIPS Proc., Vol. 44., 187-201 (1975).

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Kleinrock, L. and Naylor, W. E.

A study of line overhead in the ARPANET. Commun. ACM, Vol. 19, No. 1, 3-13 (Jan. 1976). (INWG Protocol Note 71, Sept. 1974).

Kopf, John

TYMNET as a multiplexed packet network. AFIPS Conf. Proc., Vol. 46, 609-13 (1977).

Kuo, F. F. and Abramson, N.

Some advances in radio communications for computers. IN: Proc. COMPCON73, San Francisco, Calif., Feb.-Mar. 1973,. pp 57-60 (IEEE 73CHO716-1C).

Lam, S. S.

Packet-switching in a multi-access broadcast channel with application to satellite communication in a computer network. PH.D. Thesis, UCLA-ENG-7429, Computer Science Dept., Univ. of Calif., Los Angeles, Apr. 1974.

Lam, S. S. and Kleinrock, L.

Packet switching in a multi-access broadcast channel: Performance evaluation. IEEE Trans. Commun., Vol. COM-23, 410-23 (Apr. 1975).

Lay, W. M., Mills, P. L. and Zelkowitz, M. V.

Design of a distributed computer network for resource sharing. IN: Proc. AIAA Computer Network Systems Conf., Paper 73-426, Huntsville, Alabama, Apr. 1973.

Lay, W. M., Mills, P. L. and Zelkowitz, M. V.

Operating systems architecture for a distributed computer network. IN: ACM Conference on Trends and Applications of Minicomputer Networks, Gaithersburg, Maryland, Apr. 1974.

Le Lann, G.

A general model for networks, application to ARPANET and CYCLADES. Unpublished notes, Oct. 1973.

Le Lann, G. and Le Goff, H. .

Advances in performance evaluation of communication protocols. IN: Proc. Third International Conf. Computer Comm., Trans-Canada Telephone System, Aug. 1975. pp 361-6.

LeMoli, G.

A proposal for fragmenting packets in internetworking. INWG Protocol Note 21, Apr. 1975.

Levin, J. A. and Moore, J. A.

Dialogue games: Meta-communication structures for natural language interaction. Information Sciences Institute, Univ. of Southern Calif., Los Angeles, Jan. 1976.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Lloyd, D. and Kirstein, P. T.

Alternative approaches to the interconnection of computer networks. IN: Proc. European Computing Conf. Commun. Networks, Online Conferences Ltd., Uxbridge, England, Sept. 1975. pp 25-34.

Luther, W.

Conceptual bases of Cybernet. IN: Rustin and Randal, Computer Networks, Prentice-Hall, Inc., New Jersey, 1972. pp 111-46.

MacPherson, S.

Definitions for packet-mode terms used in proposed recommendations. CCITT Study Group VII, Paper No. 212-E, Annex 9, June 1975. (INWG Protocol Note 27).

Mader, E. R., Plummer, W. W., and Tomlinson, R. S.

A protocol experiment. INWG Experiments Note 1, Aug. 1974.

Maier, M.

Out of sequence problem in a packet switching network: A simulation approach. IN: EUROCOMP, Brunel Univ., Sept. 1975. pp 191-206.

Mann, D. W.

The state of the art in the evolution of private data communication networks. IN: EUROCOMP, Brunel Univ., Sept. 1975. pp 415-32.

Mann, W. F., Ornstein, S. M. and Kraley, M. F.

A network-oriented multiprocessor front-end handling many hosts and hundreds of terminals. AFIPS Proc., Vol. 45, 533-40 (1976).

Martel, C. C., Cunningham, I. M. and Grushchow, M. S.

The BNR network: A Canadian experience with packet switching technology. Proc. IFIP Congress 74, Vol. 1, 160-4 (Dec. 1974).

McClary, W. S.

Autodin II basic architecture. IN: NTC'77 Conference Record, Vol. 3, IEEE, Inc., New York, N.Y., 1977. pp 37:2,1-2 (IEEE # 77CH1292-2 CSCB)

McCoy, C., Jr.

Improvements in routing for packet-switched networks. Interim Rept., Naval Research Lab., Washington, D.C., Feb. 18, 1975. 125 pp. (AD-A066522SL)

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

McKay, D. B. and Karp, D. P.

A network/440 protocol concept. Thomas J. Watson Research Center, International Business Machines Corp., Yorktown Heights, New York, July 1971. 15 pp.

McKay, D. B. and Karp, D. P.

Protocol for a computer network. IBM System J., Vol. 12, No. 1, 94-105 (1973).

McKenzie, A. M.

Host/Host protocol design considerations. International Network Working Group, Subgroup 2, Jan. 24, 1973. (NIC-13879)

McKenzie, A. M.

Internetwork host-to-host protocol. Bolt, Beranek, and Newman, Inc., Cambridge, Mass., Dec. 1974. (INWG Protocol Note 74)

McKenzie, A. M.

Some computer network interconnection issues. IN: Proc. National Comp. Conf., 1974, AFIPS Press. pp 857-9.

McKenzie, A. M.

TELNET protocol specification. RFC 495, Bolt Beranek and Newman, Inc., Cambridge, Mass., May 1, 1973. (NIC-15371)

McQuillan, J. M.

Adaptive routing algorithms for distributed computer networks. Ph.D. thesis, BBN Rept. No. 2831, Bolt Beranek and Newman, Inc., Cambridge, Mass., May 1974. 492 pp.  
(AD-7814676GA)

McQuillan, J. M.

Design considerations for routing algorithms in computer networks. IN: Seventh Hawaii International Conf. System Sciences, Jan. 1974, pp 22-4.

McQuillan, J. M.

The evolution of message processing techniques in the ARPA network. IN: International Computer State of the Art Rept. No. 24: Network Systems and Software, Infotech, Maidenhead, Berkshire, England, 1973. pp 541-78.

McQuillan, J. M.

Routing algorithms for computer networks - a survey. IN: NTC'77 Conference Record, Vol. 2, IEEE, Inc., New York, N. Y., 1977. pp 28:1 (IEEE # 77CH1292-2 CSCB)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

McQuillan, J. M. and Walden, D. C.

Some considerations for a high performance message-based interprocess communication system. IN: ACM SIGCOMM-SIGOPS Interface Workshop on Interprocess Communications, Mar 1975. pp 77-86.

McQuillan, J. M., Crowther, W. R., Cosell, B. P.,  
Walden, D. C. and Heart, F. E.

Improvements in the design and performance of the ARPA network. AFIPS Proc., Vol. 41, 741-54 (1972). (NIC-11626)

Merlin, P. M.

A study of the recoverability of computing systems. Tech. Rept. 58, Ph.D. Thesis, Univ. of Calif., Irvine, Nov. 1974.

Merlin, P. M.

A methodology for the design and implementation of communication protocols, IEEE Trans Commun., Vol. COM-24, 614-21 (1976).

Merlin, P. M. and Farber, D. J.

Recoverability of communication protocols--implications of a theoretical study. Res. Rept. RC 5416 (23664), Watson Research Center, IBM, Yorktown Heights, N.Y., May 1975.

Merten, A. G. and Fry, J. P.

A data description language approach to file translation. IN: Proc. 1972 ACM SIGMOD Workshop on Data Description, Access and Control, A. L. Dean (Ed.), ACM, New York, 1974. pp 191-206.

Metcalfe, R. M.

Packet communication. MAC TR-114, Project MAC, Mass. Inst. of Tech., Cambridge, Dec. 1973.

Metcalfe, R. M.

Strategies for interprocess communication in a distributed computing system. Proc. Symp. Comp. Commun. Networks and Teletraffic, Microwave Research Institute, Polytechnic Institute of Brooklyn, Apr. 1972. pp 519-26.

Metcalfe, R. M.

Strategies for operating systems in computer networks. IN: Proc. ACM Annual Conf., Vol. 1, Boston, Mass., Aug. 1972. pp 278-81.

Metzger, J. J.

A Study of an efficient retransmission strategy for data links. NTC'77 Conference Record, Vol. 1, IEEE, Inc., New York, N. Y., 1977. p 3B:1 (IEEE 77CH1292-2 CSCB)

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Michener, J. C.

Graphics protocol - level 0 only. RFC 292, Dynamic Modeling/Computer Graphics System, Mass. Institute of Technology, Cambridge, Mass., Jan. 12, 1972. 9 pp. (NIC-08302)

Millar, J. Z.

DCS AUTODIN trunking transmission between switching centers. IN: Proc. Invit. Workshop Networks of Computers, NOC-68, National Security Agency, Fort Meade, Md., Oct., 1968. pp 221-24.

Mills, P. L.

An overview of the distributed computer network. AFIPS Proc., Vol. 45, 523-31 (1976).

Mills, P. L.

Dynamic file access in a distributed computer network. TR-415, Computer Science Dept., Univ. of Maryland, Jan. 1976.

Mills, P. L.

Transient fault recovery in the distributed computer network. TR-414, Computer Science Dept., Univ. of Maryland, Jan. 1976. (Condensed version in: Proc. NBS IEEE Trends and Applications 1976 Symposium.)

Mills, P. L., Lay W. M. and Pollizzi, J.

The virtual operating system for the Distributed Computer Network. Computer Science Dept., Univ. of Maryland.

Miyahara, H., Hasegawa, T., and Teshigawara, Y.

A comparative evaluation of switching methods in computer communication networks. Proc. ICC 75, San Francisco, Calif., June 16-18, 1975. pp 6:6-10.

Myer, T. H.

Message transmission protocol. RFC 680, Bolt, Beranek, and Newman, Inc., Cambridge, Mass., 1975. (NIC-23116)

Nakamura, R.

Some design aspects of a public packet switched network. IN: Proc. ICCC, Aug 76.

Naylor, W. E.

A Loop-free adaptive routing algorithm for packet switched networks. IN: Proc. Fourth Data Commun. Symp., Quebec City, Canada, Oct. 1975. pp 7:9-14. (IEEE 75 CH1001-7 DATA)

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Neumann, A. J.

User Procedures Standardization for Network Access. Tech. Note 799, National Bureau of Standards, Washington, D.C., Oct 1973.

Neumann, Peter G.

System design for computer networks. IN: Abramson, N. and Kuo, F. F., Computer Communication Networks. Prentice-Hall, Englewood Cliffs, N. J., 1973. pp 29-81.

Newkirk, J., Kraley, M., Postel, J. and Crocker, S.

A prototypical implementation of the NCP. RFC 055, Harvard Univ., Cambridge, Mass., June 19, 1970. 32 pp (NIC-04757)

O'Sullivan, T. C.

Discussion of TELNET protocol. RFC 139, ARPA Network Information Center (NIC), SRI International, Menlo Park, Calif., May 7, 1971. 11 pp. (NIC-6717)

O'Sullivan, T. C., Metcalfe, R. M. et al.

TELNET protocol - a proposed document. RFC 158, ARPA Network Information Center (NIC), SRI International, Menlo Park, Calif., May 19, 1971. 6 pp. (NIC-06768)

Opderbeck, H. and Kleinrock, L.

The influence of control procedures on the performance of packet-switched networks. IN: National Telecommun. Conf., San Diego, Dec. 1974. pp 810-17. (INWG Note #62, Sept. 1974).

Ornstein, S. M. and Walden, D. C.

The evolution of a high performance modular packet-switch. IN: 1975 International Conf. Commun., June 1975. pp 6-17 to 6-21.

Ornstein, S. M., Crowther, W. R., Kraley, M. F., Bressler, R. D., Michel, A. and Heart, F. E.

Pluribus--A reliable multiprocessor. AFIPS Proc., Vol. 44, 551-9 (1975).

Ornstein, S. M., Heart, F. E., Crowther, W. R., Rising, H. K., Russel, S. B. and Michel, A.

The terminal IMP for the ARPA computer network. AFIPS Proc., Vol. 40, 243-54 (1972). (NIC-08218)

Padlipsky, M. A.

A proposed protocol for connecting host computers to ARPA-like networks via directly-connected front-end processors. RFC 672, The MITRE Corp., McLean, Va., Oct., 1974. (NIC-31117)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Paker, Y., Cain, G., and Morse, P.

Proc. Minicomputer interfacing. Symp. Proc., Miniconsult, London, England, 1973. 195 pp.

Pandya, R. N.

Delay analysis for DATAPAC - A packet switched network with two priority classes. pp 3:14-21 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Paoletti, L. M.

AUTODIN. IN: Grimsdale, R. L. and Kuo, F. F. (Eds.), Computer Communication Networks, Noordhoff Internat'l. Publ., Leyden, The Netherlands, 1975. pp 345-72.

Passafiume, J. J. and Wecker, S.

Distributed file access in DECnet. IN: Proc. Second Berkeley Workshop on Distributed Data Management and Computer Networks, May 25-7, 1977, Lawrence Berkeley Lab., Univ. of Calif., Berkeley, 1977. pp 114-30.

Peck, P. L.

Effective corporate networking, organization, and standardization. AFIPS Proc., Vol. 39, 561-9 (1971).

Pierce, J. R.

Network for block switching of data. Bell System Tech. J., Vol. 51, No. 6, 1133-45 (1972).

Poncet, F. and Tucker, J. B.

The design of the packet switched network for the EIN project. IN: EUROCOMP, Brunel Univ., Sept. 1975. pp 301-14.

Postel, J. B.

An informal comparison of NCP and BTF. INWG Protocol Note # 80, Aug. 1977. 24 pp.

Postel, J. B.

An informal comparison of NCP and TCP. INWG Protocol Note # 79, Aug. 1977. 24 pp.

Postel, J. B.

An informal comparison of Telenet and SMVT. INWG Protocol Note # 81, Aug. 1977. 24 pp.

Postel, J. B.

AD-hoc TELNET protocol. RFC 318, School of Engineering and Applied Science, Univ. of Calif., Los Angeles Calif., Apr. 3, 1972. (NIC-09348)

**BIBLIOGRAPHY**

**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Postel, J. B.

A graph model analysis of computer communications protocols. Ph.D. Thesis, UCLA-ENG-7410, School of Engineering and Applied Science, Univ. of Calif., Los Angeles, Jan. 1974. 193 pp. (AD-7775067GA)

Postel, J. B.

National software works protocols, Version 2. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan 1, 1975. (NIC-24856)

Postel, J. B.

Survey of network control programs in the ARPA computer network. Tech. Rept. 6722, MITRE Corp., Washington, D. C., Oct. 1974, Revision 1. 89 pp.

Pouzin, L.

A proposal for interconnecting packet switching networks. IN: EUROCOMP, Brunel Univ., May 1974. pp 1023-36.

Pouzin, L.

Basic elements of a network data link control procedure (NDLC). ACM Comp. Commun. Rev., Vol. 5, No. 1, 6-23 (Jan. 1975). (INWG Note #54) (NIC-30375)

Pouzin, L.

CIGALE, the packet switching machine of the Cyclades computer network. Proc. IFIP Congress 74, Vol. 1, 155-9 (Dec. 1974).

Pouzin, L.

Efficency of full-duplex synchronous data link procedures. INWG Protocol Note 35, June 1973. (NIC 18255)

Pouzin, L.

Interconnection of Packet Switching Networks. ALOHA System Tech. Rept. CN74-21, Univ. of Hawaii, Honolulu, Jan. 1974. (INWG Protocol Note 42, Oct. 1973)

Pouzin, L.

Network protocols. IN: D. Bates, Network Systems and Software, Infotech State of the Art Rept. 24, Infotech Information Limited, Berkshire, England, 1975. pp 599-627.

Pouzin, L.

Presentation and major design aspects of the Cyclades Computer Network. IN: Proc. 3rd Data Communications Symposium, Tampa, Fla., Nov. 1973. pp 80-85. (INWG Protocol Note 36). (NIC 18256)

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Pouzin, L.

A restructuring of X25 into HDLC, ACM Computer Commun. Rev., 9-28 (Jan. 1977).

Pouzin, L.

Standards in data communications and computer networks. IN: 4th Data Communication Symposium, Oct. 1975. pp 2.8-2.12.

Pouzin, L.

Virtual call issues in network architectures. IN: EUROCOMP, Brunel Univ., Sep. 1975. pp 603-18.

Pouzin, L.

Virtual circuits vs. datagrams - Technical and political problems. AFIPS Proc., Vol. 45, 483-93 (1976).

Price, W. L.

Simulation studies of an isarithmically controlled store and forward data communication network. IN: IFIP Congress 74, Vol. 1, North-Holland Publ. Co. Amsterdam & London, Dec. 1974. pp 151-4.

Probst, W. G. and Bochmann, G. V.

Operating system design with computer network communication protocols. pp 4:19-25 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Prosser, R. T.

Routing procedures in communications networks--Part one: Random procedures. Inst. of Radio Engineers Trans. Commun. Systems, Dec. 1962. pp 322-9.

Prosser, R. T.

Routing procedures in communications networks--Part two: Directory procedures. Inst. of Radio Engineers Trans. Commun. Systems, Dec. 1962. pp 329-35.

Pyke, T., Jr.

Computer networking technology--a state of the art review. IEEE Computer, Vol.6, No. 8, 12-19 (Aug. 1973).

Raubold, E.

An architectural approach to X25. INWG Protocol Note 52, Aug. 6, 1976. 19 pp.

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Repton, C. S. and Poncet, F.

The EIN communication subnetwork: Principles and practice. Proc. ICCC, Toronto, Canada, 1976.

Rettberg, R. D.

A brief simulation of the dynamics of an ALOHA system with slots. Bolt Beranek and Newman, Inc., Cambridge, Mass., July 31, 1972. 10 pp. (NIC-11293)

Rettberg, R. D.

Random ALOHA with slots - excess capacity. Bolt Beranek and Newman, Cambridge, Mass., Oct. 11, 1972. 6 pp. (NIC-11865)

Rettberg, R. D.

Terminal interface message processor: specifications for the interconnection of terminals and the terminal IMP. Bolt Beranek and Newman, Canoga Park, Calif., Sep. 1972. 47 pp. (AD-7769953GH)

Rettberg, R. D. and Walden, D. C.

A proposed experiment in packet broadcast satellite communications. Bolt Beranek and Newman, Cambridge, Mass., Sep. 1974.

Retz, D. L.

Operating system design considerations for the packet-switching environment. AFIPS Proc., Vol. 44, 155-60 (1975).

Ricci, F. J.

The state of the art and control of packet switches and networks. IN: Proc. IEEE 1975 Internatl. Conf. Commun., San Francisco, Calif., 1975.

Rinde, J.

Routing and control in a centrally directed Network. AFIPS Proc., Vol. 46, 603-8 (1977).

Rinde, J.

TYMNET I -- An alternative to packet switching technology. IN: Third International Conference on Computer Communications, Aug. 1976.

Roberts, L. G.

ALOHA packet system with and without slots and capture. Advanced Research Projects Agency, Washington, D.C., June 26, 1972. 13 pp. (NIC-11290)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Roberts, L. G.

Data by the packet. IEEE Spectrum, 46-51 (Feb 1974).

Roberts, L. G.

Multiple computer networks and intercomputer communication.

Commun. ACM, Vol. 11, No. 5, 296 (May 1968). (NIC-04153)

Roberts, L. G.

Packet network design--the third generation. IN: Gilchrist, B. (Ed.), Information Processing 77, IFIP, North-Holland Publishing Company, 1977. pp 541-6

Roberts, L. G. and Wessler, B. D.

Computer network development to achieve resource sharing. AFIPS Proc., Vol. 36, 543-9 (1970).

Roberts, L. G. and Wessler, B. D.

The Arpa computer network. IN: N. Abramson and F. F. Kuo, Computer-communication Networks, Prentice-Hall, Englewood Cliffs, New Jersey, 1972. pp 485-500.

Roberts, L. G.,

Extensions of packet communication technology to a hand held personal terminal. AFIPS Proc., Volume 40, 295-298 (May 16, 1972).

Rosenblum, S. R.

Progress in control procedure standardization. IN: ACM/IEEE Second Symposium on Problems in the Optimization of Data Communication Systems, 1971. pp 153-9.

Rosner, R. D.

A digital data network concept for the Defense Communications system. IN: Proc. Natl. Telecommun. Conf., Atlanta, Nov. 1973. pp 22:C1-6.

Rowe, L. A.

The distributed computing system. Tech. Rept. 66, Dept. of Information and Computer Science, Univ. of Calif., Irvine, June 1975.

Rowe, L. A., Hopwood, M. D. and Farber, D. J.

Software methods for achieving fail-soft behavior in the distributed computing system. IN: Proc. IEEE Symp. Computer Software Reliability, New York, April/May 1973, pp 7-11.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Rybaczynski, A.

Some observations on the implementation of extendable communication protocols. INWG Protocol Note 77, Aug. 1977. 4 pp.

Rybaczynski, A. M. and Weir, D. F.

Datapac X.25 service characteristics. pp 4:50-57 IN: Fifth Data Commun. Symp., ACM/IEEE, Snowbird, Utah, Sept. 27-9, 1977. (IEEE 77CH1260-9C)

Rybaczynski, A. M., Wessler, B., Despres, R. and Wedlake, J.

A new communication protocol for processing data networks--the international packet-mode interface. AFIPS Proc., Vol. 45, 477-82 (1976).

Sammes, A. J. and Winett, J. W.

An interface to the ARPA network for the CP/CMS time-sharing system. Tech. Note 1973-50, Lincoln Lab., Mass. Inst. of Tech., Cambridge, Nov. 1973.

Scantlebury, R. A. and Wilkinson, P. T.

The National Physical Laboratory data communication network. IN: Proc. ICCC Stockholm 1974. pp 223-228.

Schaft, George E.

The AUTODIN II network control center. IN: NTC'77 Conference Record, Vol. 3, IEEE, Inc., New York, N. Y., 1977. p 37:5 (IEEE 77CH1292-2 CSCB)

Schantz, R. E.

A note on reconnection protocol. RFC 671, Bolt Beranek and Newman, Cambridge, Mass., Dec. 6, 1974. (NIC-31439)

Schicker, P.

Protocol identification. European Informatics Network, Swiss Federal Institute of Technology, Zurich, May 1977. 12 pp. (INWG Protocol Note 73)

Schicker, P.

Protocols in the European Informatics Network. European Informatics Network, Swiss Federal Institute of Technology, Zurich, Nov. 1976. 5 pp. (INWG Protocol Note 60)

Schicker, P. and Duenki, A.

Bulk transfer function. European Informatics Network, Swiss Federal Institute of Technology, Zurich.

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Schicker, P. and Duenki, A.

Network job control and its supporting services. Proc. ICCC 1976.

Schicker, P. and Duenki, A.

Virtual terminal definition and protocol. European Informatics Network, Swiss Federal Institute of Technology, Zurich. 8 pp. (INWG Protocol Note #51, July 1976).

Schicker, P. and Zimmermann, H.

Proposal for a scroll mode virtual terminal in European Informatics Network., July 1976. 17 pp. (INWG Protocol Note 54)

Schneider, G. M.

DSCL--A data specification and conversion language for networks. IN: Proc. ACM SIGMOD Conf., San Jose, Calif., May 1975. pp 139-48.

Schneider, G. M.

The design and implementation of a data conversion language for networks. Ph.D. Thesis, Univ. of Wisconsin, Madison, Aug. 1974.

Schwartz, M.

Computer-communication network design and analysis. Prentice-Hall, Englewood Cliffs, N. J., 1977. 372 pp.

Schwartz, M., Boorstyn, R. R. and Prickholtz, R. L.

Terminal-oriented computer-communication networks. Proc. IEEE, 1408-23 (Nov. 1972).

Serracchioli, F.

European status of data communication. IN: Webster, E., Data communications and business systems, International Business Forms Industries/PIA, Arlington, Va., 1972. pp 153-60.

Sevcik, P. J.

AUTODIN II subscriber access protocols and interfaces. IN: NTC'77 Conference Record, Vol. 3, IEEE, Inc., New York, N. Y., 1977. p 37:6 (IEEE # 77CH1292-2 CSCB)

Sexton, J.

IIASA data link communication network data link control procedure. INWG Protocol Note 75, Feb. 1977. 68 pp.

Shu, Nan C., Lum V. Y. and Housel, B. C.

An approach to data migration in computer networks. Res. Rept. RJ1703, Watson Research Center, IBM, Yorktown Heights, N.Y., 1976.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Smetanka, T. D. and Reed, M. A.

Error considerations in determining satellite data link response time. IN: NTC'77 Conference Record, Vol. 1, IEEE , New York, N. Y., 1977. p 3B:2 (IEEE 77CH1292-2 CSCB)

Smith, D. P.

A method for data translation using the stored data definition and translation task group languages. IN: Proc. 1972 ACM SIGFIDET Workshop on Data Description, Access and Control, ACM, New York, 1972. pp 107-24.

Spier, M. and Organick, E.

The MULTICS interprocess communication facility. IN: Proc. ACM Second Symposium on Operating System Principles, Princeton Univ., Oct. 20-22, 1969. pp 83-91.

Stenning, N. V.

A data transfer protocol, Computer Networks, Vol. 1, 99-110 (1976).

Stokes, A. V. and Higginson, P. L.

The problems of connecting hosts into ARPANET. IN: Proc. European Computing Conference on Communication Networks, Online Conferences Lt., Uxbridge, England, Sept. 1975. pp 25-34.

Stutzman, B. W.

Data communication control procedures. Computing Surveys, Vol. 4, No. 4, 197-220 (Dec. 1972).

Sundstrom, R. J.

Formal definition of IBM's systems network architecture. IN: NTC'77 Conference Record, Vol. 1, IEEE, Inc., New York, N. Y., 1977. p 3A:1 (IEEE 77CH1292-2 CSCB)

Sunshine, C. A.

Efficiency of interprocess communication protocols for computer networks. Rept. P-5614, The RAND Corp., Santa Monica, Calif., Mar. 1976.

Sunshine, C. A.

Issues in communication protocol design--formal correctness. Computer Science Dept., Stanford Univ., Calif., Oct. 1974. (INWG Protocol Note 5)

Sunshine, C. A.

Factors in interprocess communication protocol efficiency for computer networks. AFIPS Proc., Vol. 45, 571-6 (1976).

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Sunshine, C. A.

Interconnection of computer networks. To appear in Computer Networks J., Vol. 1, No. 3.

Sunshine, C. A.

Interprocess communication protocols for computer networks. Ph.D. Thesis, DSL Tech. Rept. 105, Computer Science Dept., Stanford Univ., Calif., Dec. 1975.

Sunshine, C. A.

Source routing in computer networks. INWG General Note 133. 3 pp.

Tajibnapis, W. D.

A correctness proof of a topology information maintenance protocol for a distributed computer network. Commun. ACM, Vol. 20, No. 7, 477-85 (July 1977).

Tajibnapis, W. D.

Message switching protocols in distributed computer networks. Ph.D. Thesis., MCN-0676-TR-22, MERIT Computer Network, Ann Arbor, Michigan, 1976.

Takatsuki, T., Iimura, J., Nasato, C. and Abe, M.

Packet switched network in Japan. AFIPS Proc., Vol. 46, 615-21 (1977).

Taylor, H. D.

HP - communication system. IN: NTC'77 Conference Record, Vol. 3, IEEE, Inc., New York, N. Y., 1977. pp 21:6,1-5 (IEEE 77CH1292-2 CSCB)

Teicholtz, N. A.

Digital network architecture. IN: EUROCOMP, Brunel Univ., Sept. 1975. pp 13-24.

Teitelman, W. and Kahn, R. E.

A network simulation and display program. IN: Proc. Third Annual Princeton Conf. Information Sciences and Systems, Mar. 1969, 29 pp.

Thomas, R. H.

A resource-sharing executive for the ARPANET. Tech. Rept., Bolt Beranek and Newman, Cambridge, Mass., Mar. 1973. 43 pp. (AD-758162)

Thomas, R. H.

On the design of a resource sharing executive for the ARPANET. AFIPS Proc., Vol. 42, 155-63 (1973).

## BIBLIOGRAPHY

### ARPANET PROTOCOL HANDBOOK, NIC 7104

Thomas, R. H.

Reconnection protocol. RFC 426, Bolt Beranek and Newman, Cambridge, Mass., Jan. 26, 1973. (NIC-13011)

Thomas, R. H. and Clements, R. C.

FTP server-server interaction. RFC 438, Bolt Beranek and Newman, Inc., Cambridge, Mass., Jan. 15, 1973. (NIC-13770)

Tobagi, F. A. and Kleinrock, L.

Packet switching in radio channels: Part II--The hidden terminal problem in carrier sense multiple access and the busy tone solution. IEEE Trans. Commun., 1417-33 (Dec. 1975).

Tobagi, F. A., Lieberson, S. E. and Kleinrock, L.

On measurement facilities in packet radio systems. AFIPS Proc., Vol. 45, 589-96 (1976).

Tomlinson, R. S.

Selecting sequence numbers. ACM Operating Syst. Review, Vol. 9, No. 3 (July 1975). (INWG Protocol Note 2, Aug. 1974)

Toshiharu, T., Iimura, J., Chiba, M. and Masayuki, A.

Packet switched network in Japan. AFIPS Proc., Vol. 46, 615-621 (1977).

Townsend, R. L. and Watts, R. N.

Effectiveness of error control in data communications over the switched telephone network. Bell System Tech. J., Vol. 43, No. 6 (Nov. 1964).

Trafton, P. H., Blank, H. A. and McAllister, N. F.

Data transmission network computer-to-computer study. IN: Proc. Second ACM/IEEE Symp. Problems in the Optimization of Data Commun., Palo Alto, Calif., Oct. 1971. pp 183-91.

Tripathi, P. C.

Design considerations for the Menehune-Kahuna interface for the ALOHA system. A preliminary report, Univ. of Hawaii, Honolulu, Aug. 1969. 7 pp

Tugender, R. and Oestreicher, D. R.

Basic functional capabilities for a military message processing service. ISI/RR-75-23, Information Sciences Institute, Univ. Southern Calif., Marina del Rey, Calif., May 1975.

Twyver, D. A. and Rybczynski, A. M.

Datapac subscriber interfaces. IN: Proc. ICCC, Toronto, Canada, Aug. 1976. pp 143-9.

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Tymann, B.

Computer-to-computer communications systems. Computer, Vol. 6, No. 2, 27-31 (Feb. 1973).

Tymes, L.

TYMNET--A terminal oriented communications network. AFIPS Proc., Vol. 38, 211-16 (Spring 1971).

Walden, D. C.

A system for interprocess communication in a resource sharing computer network. Commun. ACM, Vol. 15, No. 4, 221-30 (Apr. 1972).

Walden, D. C.

Experiences in building, operating and using the ARPA network. IN: 2nd USA-JAPAN Computer Conf., Aug. 1975. pp 453-8.

Walden, D. C.

Host-to-host-protocols. IN: International Computer State of the Art Rept. No. 24: Network Systems and Software, Infotech Information, Maidenhead, Berkshire, England (1973). pp 287-316

Weber, J. H.

A simulation study of routing and control in communiation networks. Bell System Tech. J., Vol. 43, No. 6 (Nov. 1964).

Weiner, P.

The file transmission problem. AFIPS Proc., Vol. 42, 453 (1973).

Wessler, B. D. and Hovey, R. B.

Public packet-switched networks. Datamation, Vol. 20, No. 7, 85-7 (July 1974).

White, G. W.

Message format principles. IN: ACM/IEEE Second Symp. Problems in the Optimization of Data Commun. Systems, 1971. pp 192-8.

White, J. E.

A high-level framework for network-based resource sharing. AFIPS Proc., Vol. 45, 561-70 (1976).

White, J. E.

An NCP for the ARPA network. Computer Research Lab., Univ. of Calif., Santa Barbara, Dec. 1970. (NIC-05480)

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

White, J. E.

DPS-10 Version 2.5 implementer's guide. Augmentation Research Center, SRI International, Menlo Park, Calif., Aug 15, 1975. (NIC 26282)

White, J. E.

DPS-10 Version 2.5 Programmer's Guide. Augmentation Research Center, SRI International, Menlo Park, Calif., Aug. 13, 1975. (NIC 26271)

White, J. E.

DPS-10 Version 2.5 Source Code. Augmentation Research Center, SRI International, Menlo Park, Calif., Aug. 13, 1975. (NIC 26267)

White, J. E.

Elements of a distributed programming system. J. Computer Languages, 1976.

White, J. E.

Network specifications for remote job entry and remote job output retrieval at UCSB. RFC 105, Computer Science Dept., Univ. of Calif., Santa Barbara, Calif., Mar. 22, 1971. (NIC-05775)

White, J. E.

Procedure call protocol specification. Augmentation Research Center, SRI International, Menlo Park, Calif., Nov. 1974.

White, J. E.

The procedure call protocol, version 2. Augmentation Research Center, SRI International, Menlo Park, Calif., Jan. 1, 1975. (NIC-24855)

Whitney, V. K. M.

A study of optimal file assignment and communication network configuration. Ph.D. thesis., Univ. of Michigan, Ann Arbor, 1970.

Winett, J. M. and Sammes, A. J.

An interface to the ARPA network for the CP/CMS timesharing system. Vols. 1 and 2, Lincoln Lab., Mass. Institute of Tech., Cambridge, Nov. 1973.

Wolfe, E. W.

An advanced computer communication network. IN: AIAA Computer Network Systems Conference, Paper no. 73-414, Apr. 1973.

BIBLIOGRAPHY  
ARPANET PROTOCOL HANDBOOK, NIC 7104

Yamaguchi, K. and Merten, A. G.

Methodology for transferring programs and data. IN: Proc. ACM/SIGMOD Workshop on Data Description, Access and Control, ACM, New York, 1974. pp 141-56.

Young, S. C. K. and McGibbon, C. I.

The control system of the Datapac Network. IN: Proc. ICCC, Toronto, Aug. 1976. pp 137-42.

Zimmerman, H.

The CYCLADES end-to-end protocol. IN: IEEE Fourth Data Communication Symposium, Oct 1975. pp 7.21-26. (IEEE 75 CH1001-7 DATA)

Zimmerman, H. and Elie, M.

Transport protocol. Standard host-host protocol for heterogeneous computer networks, June 1973. (INWG Protocol Note 61)

Zimmermann, H.

High level protocols standardization: Technical and political issues. Proc. Third ICCC., 373-6 (Aug. 1976).

Zimmermann, H.

Protocol design and specification. IFIP WG 6.1 Study Group B, Rept. from the Toronto Working Party on VTP, Aug. 1977. (INWG Protocol Note 82, Aug. 1977). 24 pp.

No author

Basic Message Format for Inter-network Communication. IFIP-WG6.1, INWG Note 83, May 1975. 7 pp.

CCCIT-COM VII--Proposal for revised draft recommendation X25. France and UK Post-Office, Nov 1975. 67 pp.

CSTS Concepts and Capabilities. Report E000175-01, Revision 1, Infonet Div., Computer Sciences Corp., El Segundo, Calif., March 1973.

Datapac standard network access protocol specification. Trans-Canada Telephone System, Mar. 1976.

Datapac standard network access protocol. Trans-Canada Telephone System, Ottawa, Canada, 1974.

Digital network architecture: Network services protocol. Digital Equipment Corp., Marlboro, Mass., July 1975. 44 pp.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

EIN Reports and Standards, EIN/EB/750123, National Physical Laboratory, Teddington, England.

Encryption algorithm for computer data protection. National Bureau of Standards, Federal Register, Vol. 40, No. 52 (Mar 1975).

End-to-End Protocol (Specifications), EIN/76/003, National Physical Laboratory, Teddington, England.

End-to-End Protocol for EIN. EIN Report No. EIN/76/002, National Physical Laboratory, Teddington, England.

IBM 3600 finance communication system--System summery. Rept. No. GC27-0001-2, Watson Research Center, IBM, Yorktown Heights, N.Y., 1973.

IBM synchronous data link control--General information. Rept. No. GA27-3093-0, Watson Research Center, IBM, Yorktown Heights, N.Y., 1974.

IBM system network architecture, general information, GA 27-3102-0. Jan. 1975. 48pp.

IFIP WG6.1--Proposal for an internetwork end-to-end protocol. INWG Protocol Note 96, Sept 1975. 29 pp.

Interface message processors for the ARPA computer network, QTR No. 1. BBN Rept. No. 3063, Bolt Beranek and Newman, Cambridge, Mass., April 75.

Interface message processors for the ARPA computer network, QTR No. 2. BBN Rept. No. 3106, Bolt Beranek and Newman, Cambridge, Mass., July 75.

ISO/TC97/SC6, Documents 1145, 1167, 1173, 1205, 1249, 1258, 4335. Proposed Draft International Standard on Elements of Procedures. (Contain items on the structure of network protocols).

ISO, HDLC proposed balanced class of procedures, TC97/SC6/NI340, Nov. 1976. 11 pp.

A network independent file transfer protocol, High Level Protocol Group, Computer Lab., Univ. of Cambridge, England., 1977. (INWG Protocol Note 86)

NIC/ARPANET protocol analysis (System E). IN: Neumann, A. J., User Procedures Standardization for Network Access, Tech. Note 799, National Bureau of Standards, Washington, D.C., Oct 73. 27 pp.

Proposal for revised draft recommendation X25. CCITT Study Group VII.

Recommendation X.25. Telenet Communications Corporation, Washington, D. C., Apr. 1976. 74 pp.

Some Considerations on flow control for international packet transport. INWG Protocol Note #8, June 1974.

**BIBLIOGRAPHY**  
**ARPANET PROTOCOL HANDBOOK, NIC 7104**

Specification of the Interface between a subscriber computer and a network switching center, packet format and operation of services, SESA-LOGICA 7104/5 2240-2001 1/02, Logica, London.

Standard network access protocol specification. The Computer Communications Group, Trans-Canada Telephone System, Ottawa, Canada, March 1976.

Systems Network Architecture--General Information. Tech. Rept. GA 27-3102-0, Watson Research Center, IBM, Yorktown Heights, N.Y., Jan. 1975. 50 pp.

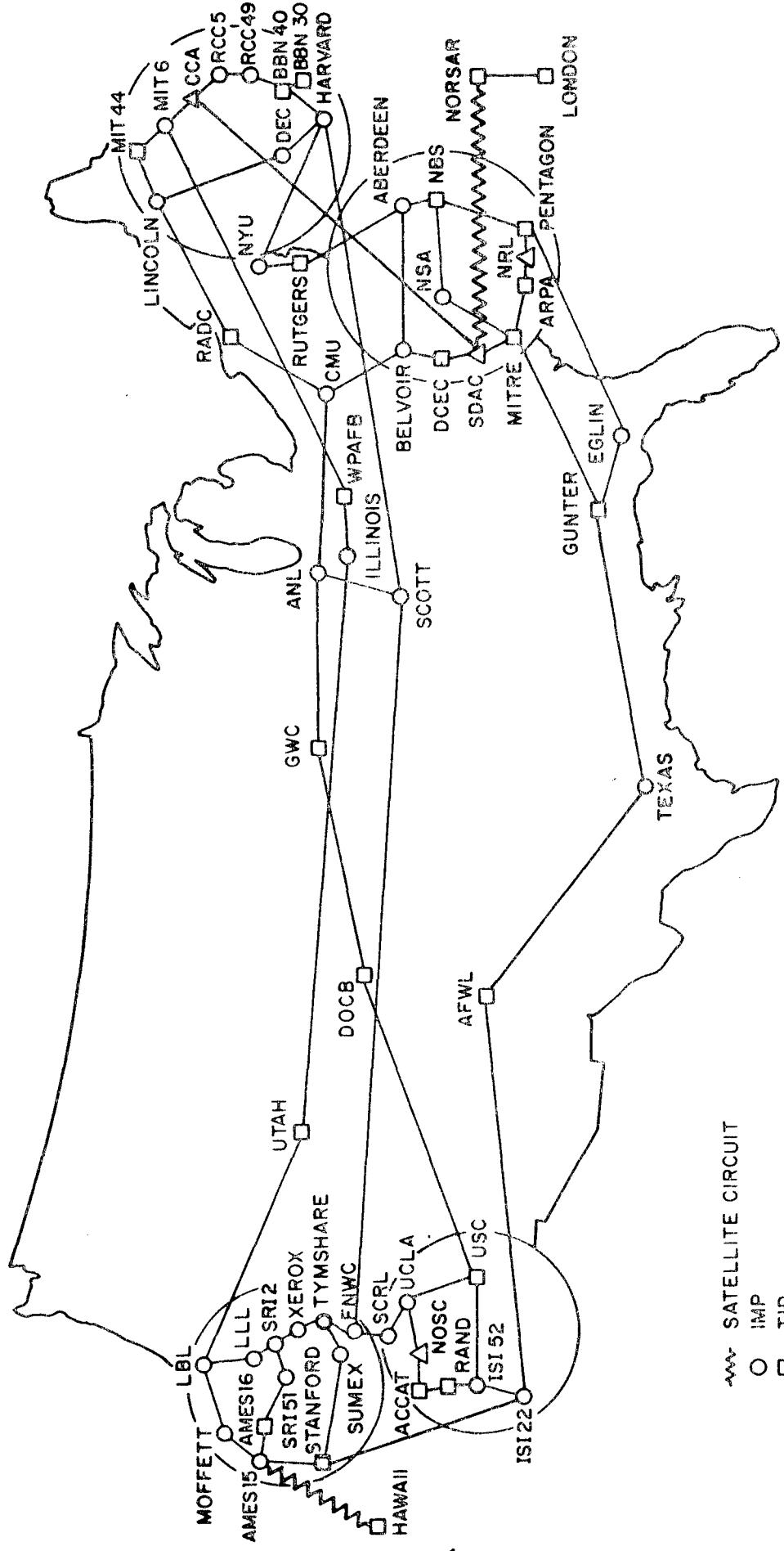
The interface message processor program. Tech. Info. Rept. 89, Bolt, Beranek and Newman, Inc., Cambridge, Mass., Mar. 1974.

The remote job entry mini-host. BBN Tech. Info. Rept. 93, Bolt Beranek and Newman, Inc., Cambridge, Mass., Aug. 1974.

TELENET Host interface specification, Telenet Communication Corp., Washington, D. C., Mar. 1975. 120 pp.



ARPANET GEOGRAPHIC MAP, NOVEMBER 1977



~~~ SATELLITE CIRCUIT  
 ○ IMP  
 □ TIP  
 △ PLURIBUS IMP

(NOTE: THIS MAP DOES NOT SHOW ARPA'S EXPERIMENTAL  
 SATELLITE CONNECTIONS)

NAMES SHOWN ARE IMP NAMES, NOT (NECESSARILY) HOST NAMES

