MIL-STD-1780
10 May 1984

MILITARY STANDARD

FILE TRANSFER PROTOCOL

NO DELIVERABLE DATA
REQUIRED BY THIS DOCUMENT

IPSC/SLHC/TCTS

DEPARTMENT OF DEFENSE
WASHINGTON, D.C. 20301

File Transfer Protocol

MIL-STD-1780

1. This Military Standard is approved for use by all Departments and Agencies of the Department of Defense.

2. Beneficial comments (recommendations, additions, deletions) and any pertinent data which may be of use in improving this document should be addressed to: Defense Communications Engineering Center, ATTN: R130, 1860 Wiehle Avenue, Reston, Virginia 22090-5500, by using the self-addressed Standardization Document Improvement Proposal (DD Form 1426) appearing at the end of this document, or by letter.

FOREWORD

This document specifies the File Transfer Protocol (FTP) which supports the transfer of file data throughout a heterogeneous host computer network. This draft standard defines the FTP's role and purpose, defines the services provided to users and specifies the mechanisms needed to support these services.

CONTENTS

Paragraph			Page
	1.	SCOPE	1
	1.1	Purpose	1
	1.2	Organization	1
	1.3	Application	1
	1.4	Objectives	1
	2.	REFERENCED DOCUMENTS	2
	2.1	Issues of documents	2
	2.2	Other publications	2
	3.	DEFINITIONS	3
	3.1	Definition of terms	3
	4.	GENERAL REQUIREMENTS	6
	4.1	General	6
	4.2	The FTP model	6
	4.2.1	TELNET initiation	7
	4.3	FTP data connection	7
	4.3.1	TELNET connections	7
	4.4	Data transfer functions	8
	5.	DATA REPRESENTATION AND STORAGE	9
	5.1	Data representation differences	9
	5.1.1	Data representation in binary	9
	5.2	Data representation types	9
	5.2.1	ASCII format	9
	5.2.2	EBCDIC format	10
	5.3	Reasons for file transfer	10
	5.3.1	File transfer parameters	10
	5.3.1.1	Non-print	10
	5.3.1.2	TELNET format controls	10
	5.3.1.3	Carriage control (ASA)	10
	5.3.1.4	Image	11
	5.3.1.5	Local byte size	11
	5.3.1.5.1	Local byte example 1	11
	5.3.1.5.2	Local byte example 2	11
	5.3.1.6	File parameter caution	11
	5.3.2	File structure	12
	5.3.2.1	File- vs. record-oriented	12
	5.3.2.2	Page structure	13
	5.4	Establishing data connections	14
	5.4.1	Passive data transfer process	14
	5.4.2	Alternate data	14
	5.5	Transmission modes	15
	5.5.1	Transmission modes defined	15

CONTENTS - Continued

			Page
Paragraph	5.5.1.1	Stream	15
	5.5.1.2	Block	16
	5.5.1.2.1	Descriptor codes	16
	5.5.1.3	Compressed	17
	5.6	Error recovery and restart	18
	5.7	File transfer functions	19
	5.7.1	FTP commands	19
	5.7.1.1	Access control commands	19
	5.7.1.1.1	User name (USER)	19
	5.7.1.1.2	Password (PASS)	19
	5.7.1.1.3	Account (ACCT)	20
	5.7.1.1.4	Reinitialize (REIN)	20
	5.7.1.1.5	Logout (QUIT)	20
	5.7.2	Transfer parameter commands	20
	5.7.2.1	Data port (PORT)	21
	5.7.2.2	Passive (PASV)	21
	5.7.2.3	Representation type (TYPE)	21
	5.7.2.4	File structure (STRU)	21
	5.7.2.5	Transfer mode (MODE)	22
	5.7.3	FTP service commands	22
	5.7.3.1	Retrieve (RETR)	22
	5.7.3.2	Store (STOR)	22
	5.7.3.3	Append (with create) (APPE)	22
	5.7.3.4	Allocate (ALLO)	22
	5.7.3.5	Restart (REST)	23
	5.7.3.6	Rename from (RNFR)	23
	5.7.3.7	Rename to (RNTO)	23
	5.7.3.8	Abort (ABOR)	23
	5.7.3.8.1	FTP service command completed	23
	5.7.3.8.2	FTP service command in progress	23
	5.7.3.10	Delete (DELE)	23
	5.7.3.11	Change working directory (CWO)	24
	5.7.3.12	List (LIST)	24
	5.7.3.13	Name-list (NLST)	24
	5.7.3.14	Site parameters (SITE)	24
	5.7.3.15	Status (STAT)	24
	5.7.3.16	Help (HELP)	24
	5.7.3.17	Noop (NOOP)	25
	5.7.4	TELNET language	25
	5.8	FTP replies	25
	5.8.1	FTP reply defined	26
	5.8.2	Three-digit reply defined	27
	5.8.2.1	First digit values	27
	5.8.2.1.1	Positive preliminary reply (1yz)	27

CONTENTS - Continued

			Page
Paragraph	5.8.2.1.2	Positive completion reply (2yz)- - - - - - -	27
	5.8.2.1.3	Positive intermediate reply (3yz)- - - - - -	27
	5.8.2.1.4	Transient negative completion reply (4yz)- -	27
	5.8.2.1.5	Permanent negative completion reply (5yz)- -	28
	5.8.2.2	Second digit values- - - - - - - - - - - - -	28
	5.8.2.2.1	Syntax (x0z) - - - - - - - - - - - - - - - -	28
	5.8.2.2.2	Information (x1z)- - - - - - - - - - - - - -	28
	5.8.2.2.3	Connections (x2z)- - - - - - - - - - - - - -	28
	5.8.2.2.4	Authentication and accounting (x3z)- - - - -	28
	5.8.2.2.5	Unspecified (x4z)- - - - - - - - - - - - - -	28
	5.8.2.2.6	File system (x5z)- - - - - - - - - - - - - -	28
	5.8.2.3	Third digit values - - - - - - - - - - - - -	28
	5.8.3	Reply codes by function groups - - - - - - -	29
	5.8.4	Numeric order list of reply codes- - - - - -	31
	5.9	Declarative specifications - - - - - - - - -	33
	5.9.1	Minimum implementation - - - - - - - - - -	33
	5.9.2	Connections- - - - - - - - - - - - - - - - -	34
	5.9.2.1	Command-reply sequence - - - - - - - - - - -	34
	5.9.3	Commands - - - - - - - - - - - - - - - - - -	35
	5.9.3.1	FTP command list - - - - - - - - - - - - - -	35
	5.9.3.2	FTP command syntax - - - - - - - - - - - - -	36
	5.10	Sequencing of commands and replies - - - - -	36
	5.10.1	Command-reply sequences- - - - - - - - - - -	37
	5.11	State diagrams - - - - - - - - - - - - - - -	40
	5.11.1	Largest group of FTP commands- - - - - - - -	40
	5.11.2	Other large group of FTP commands- - - - - -	40
	5.11.3	Rename sequence command- - - - - - - - - - -	41
	5.11.4	Restart command- - - - - - - - - - - - - - -	42
	5.11.5	Login sequence - - - - - - - - - - - - - - -	43
	5.11.6	Command and reply interchange- - - - - - - -	44
	6.	TYPICAL FTP SCENARIO- - - - - - - - - - - - -	45
	6.1	Scenario - - - - - - - - - - - - - - - - - -	45
	6.2	Connection establishment - - - - - - - - - -	45

FIGURES

			Page
Figure	1	Example FTP configuration in a host protocol hierarchy- - - - - - - - - - - - - - -	6
	2	Model for FTP use- - - - - - - - - - - - - - - -	6
	3	Server-server interaction- - - - - - - - - - -	7
	4	Block header - - - - - - - - - - - - - - - - - -	16
	5	Transmission of a six-character marker - - - -	17
	6	Byte string- - - - - - - - - - - - - - - - - -	17
	7	Replicated byte- - - - - - - - - - - - - - - -	18
	8	Filler string- - - - - - - - - - - - - - - - -	18
	9	Largest group of FTP commands- - - - - - - - -	40
	10	Other group of FTP commands- - - - - - - - - -	40
	11	Rename sequence- - - - - - - - - - - - - - - -	41
	12	Restart command- - - - - - - - - - - - - - - -	42
	13	Login sequence - - - - - - - - - - - - - - - -	43
	14	Command and reply interchange- - - - - - - - -	44

1. SCOPE

1.1 Purpose. This standard establishes criteria for the File Transfer Protocol (FTP) used for tranferring files between computer systems.

1.2 Organization. This standard introduces the File Transfer Protocol's role and purpose, defines the services provided to users, and specifies the mechanisms needed to support those services.

1.3 Application. The File Transfer Protocol is approved for implementation in hosts of all DoD packet switching networks which connect or have the potential for utilizing connectivity across network and subnetwork boundaries and which require a file transfer service. The term network as used herein includes Local Area Networks.

1.4 Objectives. 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 computer users, with a simple and easily implemented protocol design. This standard assumes knowledge of Transmission Control Protocol, MIL-STD-1778, and TELNET Protocol, MIL-STD-1782.

2. REFERENCED DOCUMENTS

2.1 Issues of documents. The following documents of the issue in effect on date of invitation for bids or request for proposal, form a part of this standard to the extent specified herein. (The provisions of this paragraph are under consideration.)

Standards:

Federal

FED-STD-1037	Glossary of Telecommunications Terms

Military

MIL-STD-1778	Transmission Control Protocol
MIL-STD-1782	TELNET Protocol

2.2 Other publications. The following documents form a part of this standard to the extent specified herein. Unless otherwise indicated, the issue in effect on date of invitation for bids or request for proposal shall apply. (The provisions of this paragraph are under consideration.)

3. DEFINITIONS

3.1 Definition of terms. The definition of terms used in this standard shall comply with FED-STD-1037. Terms and definitions unique to MIL-STD-1780 are contained herein.

a. Byte size. There are two byte sizes of interest in FTP: the logical byte size of the file, and the transfer byte size used for the transmission of the data. The transfer byte size is always 8 bits. The transfer 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.

b. Command Connection. A TELNET connection in which FTP commands and replies are exchanged between the user and server FTP processes, respectively. Although the degree to which the TELNET protocol is utilized is not specified, both user and server processes must have the capability to participate in TELNET negotiations. (The default port for the command connection is defined as Port L.)

c. Data connection. A simplex connection over which data is transferred, in a specified 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.

d. Data port. The passive data transfer process "listens" on the data port for a connection from the active transfer process in order to open the data connection.

e. EOF. The end-of-file condition that defines the end of a file being transferred.

f. EOR. The end-of-record condition that defines the end of a record being transferred.

g. 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.

h. FTP commands. A set of commands that comprise the control information flowing from the user-FTP to the server-FTP process.

i. File. An ordered set of computer data (including programs), of arbitrary length, uniquely identified by a pathname.

j. 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.

k. NVT. The Network Virtual Terminal as defined in the TELNET Protocol (paragraph 4.2.1, MIL-STD-1782.)

l. NVFS. The Network Virtual File System. A concept which defines a standard network file system with standard commands and pathname conventions. FTP only partially implements the NVFS concept at this time.

m. Page. A file may be structured as a set of independent parts called pages. FTP supports the transmission of discontinuous files as independent indexed pages.

n. 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 involved in the transfer.

o. 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.

p. Reply. A reply is an acknowledgment (positive or negative) sent from server to user via the command 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.

q. Server-DTP. The data transfer process, in its normal "active" state, establishes the data connection with the "listening" data port, 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, a connection on the data port.

r. 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).

s. Server-PI. The protocol interpreter "listens" on Port L for a connection 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.

t. TELNET connections. The full-duplex communication path between a user-PI and a server-PI, operating according to the TELNET Protocol.

u. 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.

v. User-DTP. The data transfer process "listens" on the data port for a connection from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive.

w. 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.

x. User-PI. The protocol interpreter initiates the command connection from its port U to the server-FTP process, initiates FTP commands, and governs the user-DTP if that process is part of the file transfer.

4. GENERAL REQUIREMENTS

4.1 General. In specifying a File Transfer Protocol it is desirable not to assume a set system configuration. As a practical matter, the distributions of the file transfer protocol will vary with specific hardware configurations. Although appearing to focus on FTP implementations, this standard can apply to any configuration given appropriate protocols to bridge hardware boundaries. An example of where FTP is configured in a host protocol hierarchy can be seen in Figure 1.

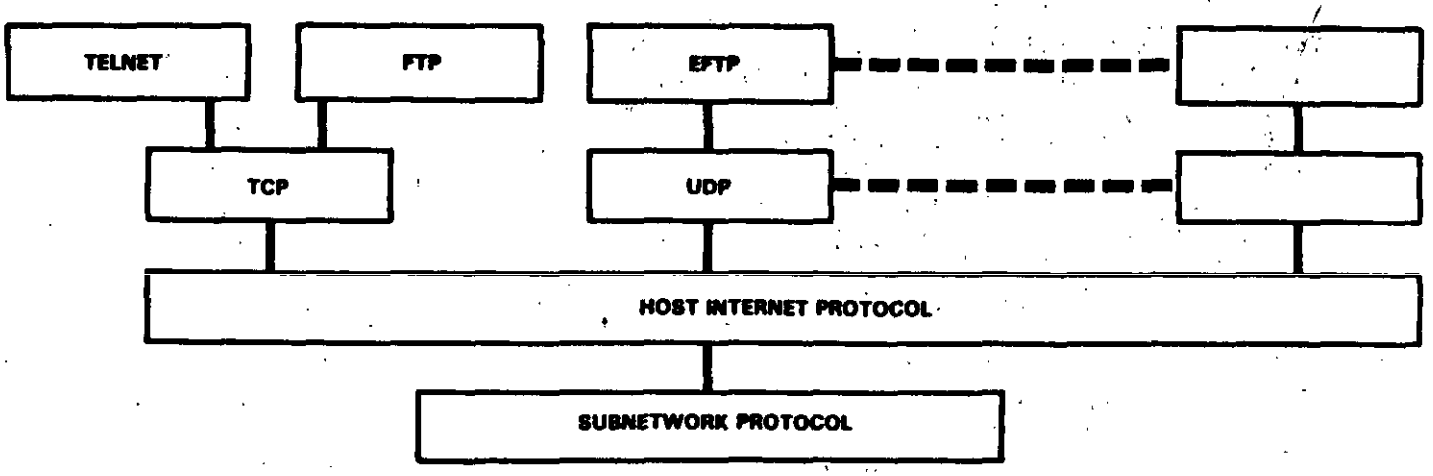
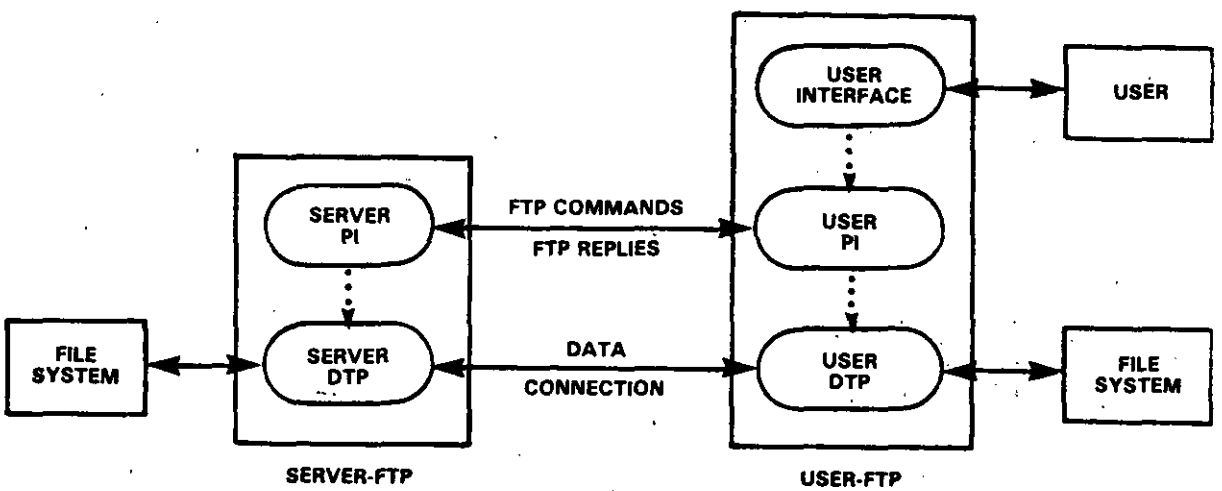


FIGURE 1. Example FTP configuration in a host protocol hierarchy.

4.2 The FTP model. The following model (see Figure 2) may be diagrammed for an FTP service.



NOTES: 1. The data connection may be used in either direction.
2. The data connection need not exist all of the time.

FIGURE 2. Model for FTP use.

4.2.1 Command initiation. In Figure 2, the user-protocol interpreter initiates the command connection. At the initiation of the user, standard FTP commands are generated by the user-PI and transmitted to the server process via the command connection. (The user may establish a direct command connection to the server-FTP, from a TAC terminal for example, and generate standard FTP commands himself, bypassing the user-FTP process.) Standard replies are sent from the server-PI to the user-PI over the command connection in response to the commands.

4.3 FTP data connections. The FTP commands specify the parameters for the data connection (data port, 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 port, and the server initiate the data connection and data transfer in accordance with the specified parameters. It should be noted that the data port need not be in the same Host that initiates the FTP commands via the command connection, but the user or the user-FTP process must ensure a "listen" on the specified data port. It should also be noted that the data connection may be used for simultaneous sending and receiving. In another situation, a user might wish to transfer files between two Hosts, neither of which is his local Host. He sets up command 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. Figure 3 is a model of this server-server interaction.

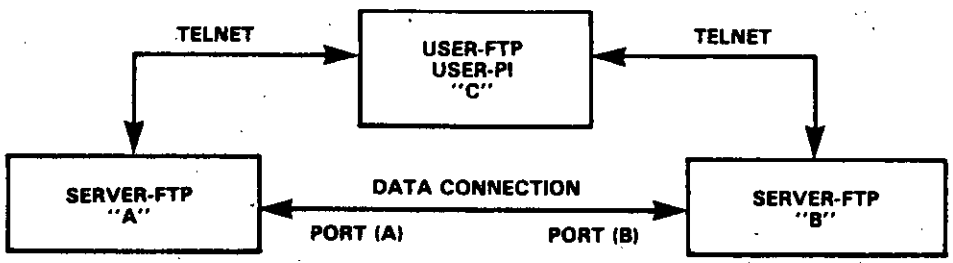


FIGURE 3. Server-server interaction.

4.3.1 Command connections. The protocol requires that the command connections be open while data transfer is in progress. It is the responsibility of the user to request the closing of the command connections when finished using the FTP service, while it is the server who takes the action. The server may abort data transfer if the command connections are closed without command.

4.4 Data transfer functions. Files are transferred only via the data connection. The command 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 MODE command 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.

5. DATA REPRESENTATION AND STORAGE

5.1 Data representation differences. 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. A computer may store NVT-ASCII as five 7-bit ASCII characters, left-justified in a 36-bit word. Another system may store NVT-ASCII as 8-bit EBCDIC codes. 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.

5.1.1 Data representation in binary. 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 limited capability should be performed by the user directly.

5.2 Data representation types. 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, called the "transfer byte size", and the two should not be confused. For example, NVT-ASCII has a logical byte size of 8 bits. If the type is Local byte, then the TYPE command has an obligatory second parameter specifying the logical byte size. The transfer byte size is always 8 bits. 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 data representation types are defined in paragraphs 5.2.1 and 5.2.2.

5.2.1 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 in MIL-STD 1782). 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.

5.2.2 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.

5.3 Reasons for file transfer. 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.

5.3.1 File transfer parameters. 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:

5.3.1.1 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.

5.3.1.2 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.

5.3.1.3 Carriage control . The file contains American National Standards Institute (ANSI) (FORTRAN) vertical format control characters. 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 ANSI 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 ANSI controls.

5.3.1.4 Image. The data are sent as contiguous bits which, for transfer, are packed into the 8-bit transfer bytes. The receiving site must store the data as contiguous bits. The structure of the storage system might necessitate the padding of 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 zeros, 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.

5.3.1.5 Local 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.

5.3.1.5.1 Local byte example 1. 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.

5.3.1.5.2 Local byte example 2. A pair of hosts with a 36-bit word size may send data to one another in words by using TYPE L 36. The data would be sent in the 8-bit transmission bytes packed so that 9 transmission bytes carried two host words.

5.3.1.6 File parameter caution. 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.

5.3.2 File structure. In addition to different representation types, FTP allows the structure of a file to be specified. Three file structures are defined in FTP:

file-structure, where there is no internal structure and the file is considered to be a continuous sequence of data bytes,

record-structure, where the file is made up of sequential records,

and page-structure, where the file is made up of independent indexed pages.

File-structure is the default, to be assumed if the STRUcture command has not been used but both file and record 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 in fixed length records, but may also be stored 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.

5.3.2.1 File- vs. record-oriented. 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.

5.3.2.2 Page structure. To transmit files that are discontinuous FTP defines a page structure. Files of this type are sometimes known as "random access files" or even as "holey files". In these files there is sometimes other information associated with the file as a whole (e.g., a file descriptor), or with a section of the file (e.g., page access controls), or both. In FTP, the sections of the file are called pages. To provide for various page sizes and associated information each page is sent with a page header. The page header has the following defined fields:

a. Header Length. The number of logical bytes in the page header including this byte. The minimum header length is 4.

b. Page Index. The logical page number of this section of the file. This is not the transmission sequence number of this page, but the index used to identify this page of the file.

c. Data Length. The number of logical bytes in the page data. The minimum data length is 0.

d. Page Type. The type of page this is. The following page types are defined:

0 = Last Page

This is used to indicate the end of a paged structured transmission. The header length must be 4, and the data length must be 0.

1 = Simple Page

This is the normal type for simple paged files with no page level associated control information. The header length must be 4.

2 = Descriptor Page

This type is used to transmit the descriptive information for the file as a whole.

3 = Access Controlled Page

This type includes an additional header field for paged files with page level access control information. The header length must be 5.

e. Optional Fields. Further header fields may be used to supply per page control information, for example, per page access control.

All fields are one logical byte in length. The logical byte size is specified by the TYPE command.

5.4 Establishing data connections. The mechanics of transferring data consists of setting up the data connection to the appropriate ports and choosing the parameters for transfer. Both the user and the server-DTPs have a default data port. The default data connection is established between user process port U and server port L-1. The transfer byte size is 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.

5.4.1 Passive data transfer process. The passive data transfer process (this may be a user-DTP or a second server-DTP) shall "listen" on the data port prior to sending a transfer request command. The FTP request command determines the direction of the data transfer. The server, upon receiving the transfer request, will initiate the data connection to the port. When the connection is established, the data transfer begins between DTP's, and the server-PI sends a confirming reply to the user-PI.

5.4.2 Alternate data. It is possible for the user to specify an alternate data port by use of the PORT command. For example, he might want a file retrieved from a third party Host. In the latter case the user-PI sets up command connections with both server-PI's. One server is then told (by an FTP command) to "listen" for a connection which the other will initiate. The user-PI sends one server-PI a PORT command indicating the data port of the other. 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 it and to close it. 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:

a. The server has completed sending data in a transfer mode that requires a close to indicate EOF.

b. The server receives an ABORT command from the user.

c. The port specification is changed by a command from the user.

d. The TELNET connection is closed legally or otherwise.

e. 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.

5.5 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. For files transmitted in page structure a "last-page" page type is used. 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 structure, and the receiving Host will perform the inverse translation to his internal denotation. A host specific 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.

NOTE: In the rest of this section, byte means "transfer byte" except where explicitly stated otherwise.

5.5.1 Transmission modes defined. The following transmission modes are defined in FTP:

5.5.1.1 Stream. The data is transmitted as a stream of bytes. There is no restriction on the representation type used; record structures are allowed. In a record structured file EOR and EOF will each be indicated by a two-byte control code. 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 zeros 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 structure is file structure, the EOF is indicated by the sending Host closing the data connection and all bytes are data bytes.

5.5.1.2 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. The header consists of the three bytes. 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 in Figure 4.

FIGURE 4. Block header.

5.5.1.2.1 Descriptor codes. 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

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). <SP> (Space, in the appropriate language) must not be used WITHIN a restart marker. For example, Figure 5 shows the transmission of a six-character marker.

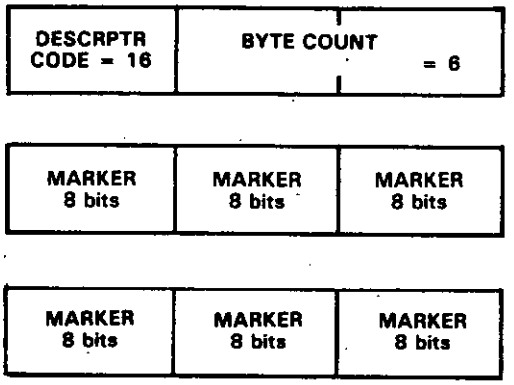
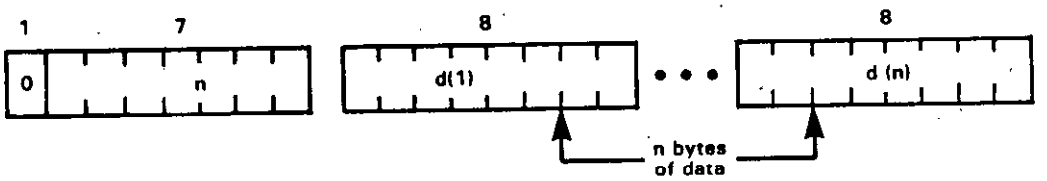


FIGURE 5. Transmission of a six-character marker.

5.5.1.3 Compressed. 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 n>0 bytes (up to 127) 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 7 bits containing the number n. (See Figure 6).



String of n data bytes d(1),..., d(n)
Count n must be positive.

FIGURE 6. Byte string.

To compress a string of n replications of the data byte d, 2 bytes are sent as shown in Figure 7.

FIGURE 7. Replicated byte.

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 type is Image or Local byte the filler is a zero byte. (See Figure 8).

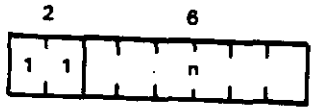


FIGURE 8. Filler string.

The escape sequence is a double byte, the first of which is the escape byte (all zeros) and the second of which contains descriptor codes as defined in Block 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 can be most effectively used to reduce the size of printer files such as those generated by RJE Hosts.

5.6 Error recovery and restart. There is no provision for detecting bits lost or scrambled in data transfer; this level of error control is handled by the TCP. However, a restart procedure is provided to protect users from gross system failures (including failures of a Host, an FTP-process, or the underlying network). 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 of printable characters in the default or negotiated language of the TELNET connection (ASCII or EBCDIC). 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 command connection in a 110 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 command connection and is immediately followed by the command (such as RETR, STOR or LIST) which was being executed when the system failure occurred.

5.7 File transfer functions. The communication channel from the user-PI to the server-PI is established by a TCP connection from the user to a standard server port. 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. The FTP replies are discussed in the next section. In the description of a few of the commands in this section it is helpful to be explicit about the possible replies.

5.7.1 FTP commands. The following are FTP commands:

5.7.1.1 Access control commands. The following commands specify access control identifiers (command codes are shown in parentheses).

5.7.1.1.1 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 command 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.

5.7.1.1.2 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.

5.7.1.1.3 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 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 332. On the other hand, if account information is NOT required for login, the reply to a successful PASSword command is 230; and if the account information is needed for a command issued later in the dialogue, the server should return a 332 or 532 reply depending on whether he stores (pending receipt of the ACCounT command) or discards the command, respectively.

5.7.1.1.4 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 command connection is left open. This is identical to the state in which a user finds himself immediately after the command connection is opened. A USER command may be expected to follow.

5.7.1.1.5 Logout (QUIT). This command terminates a USER and if file transfer is not in progress, the server closes the command 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 QUIT. An unexpected close on the command connection will cause the server to take the effective action of an abort (ABOR) and a logout (QUIT).

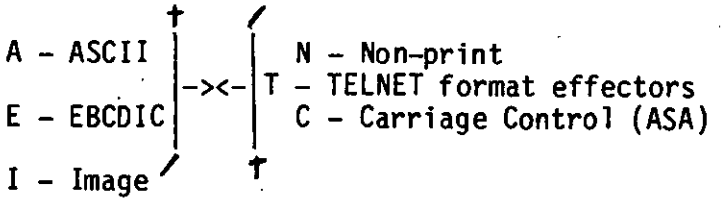
5.7.2 Transfer parameter commands. All data transfer parameters have default values, and the commands specifying data transfer parameters are required only if the default parameter values are to be changed. The default value is the last specified value, or if no value has been specified, the standard 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.

5.7.2.1 Data port (PORT). The argument is a HOST-PORT specification for the data port to be used in data connection. There defaults for both the user and server data ports, and under normal circumstances this command and its reply are not needed. If this command is used the argument is the concatenation of a 32-bit internet host address and a 16-bit TCP port address. This address information is broken into 8-bit fields and the value of each

field is transmitted as a decimal number (in character string representation). The fields are separated by commas. A port command would be PORT h1,h2,h3,h4, p1,p2; where, h1 is the high order 8 bits of the internet host address.

5.7.2.2 Passive (PASV). This command requests the server-DTP to "listen" on a data port (which is not its default data port) and to wait for a connection rather than initiate one upon receipt of a transfer command. The response to this command includes the host and port address this server is listening on.

5.7.2.3 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:



L <byte size> - Local byte Byte size

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.

5.7.2.4 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
P - Page structure

The default structure is File.

5.7.2.5 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.

5.7.3 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 command 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:

5.7.3.1 Retrieve (RETR). This command causes the server-DTP to transfer a copy of the file, specified in the pathname, to the serveror user-DTP at the other end of the data connection. The status and contents of the file at the server site shall be unaffected.

5.7.3.2 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.

5.7.3.3 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.

5.7.3.4 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 or page structure a maximum record or page 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 or page size should accept a dummy value in the first argument and ignore it.

5.7.3.5 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.

5.7.3.6 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.

5.7.3.7 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.

5.7.3.8 Abort (ABOR). This command tells 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 connection is not to be closed by the server, but the data connection must be closed. There are two cases for the server upon receipt of this command: (1) the FTP service command was already completed, or (2) the FTP service command is still in progress.

5.7.3.8.1 FTP service command completed. In the first case, the server closes the data connection (if it is open) and responds with a 226 reply, indicating that the abort command was successfully processed.

5.7.3.8.2 FTP service command in progress. In the second case, the server aborts the FTP service in progress and closes the data connection, returning a 426 reply to indicate that the service request terminated in abnormally. The server then sends a 226 reply, indicating that the abort command was successfully processed.

5.7.3.9 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 deleteμ"), it should be provided by the user-FTP process.

5.7.3.10 Change working directory (CWO). 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.

5.7.3.11 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).

5.7.3.12 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.)

5.7.3.13 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.

5.7.3.14 Status (STAT). This command shall cause a status response to be sent over the command connection in the form of a reply. The command may be sent during a file transfer (along with the command 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 command 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.

5.7.3.15 Help (HELP). This command shall cause the server to send helpful information regarding its implementation status over the command 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 211 or 214. 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.

5.7.3.16 Noop (NOOP). This command does not affect any parameters or previously entered commands. It specifies no action other than that the server send an OK reply.

5.7.4 TELNET language. The File Transfer Protocol follows the specifications of the TELNET protocol for all communications over the command connection. Since 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, QUIT) may be sent over the command connection while a data transfer is in progress. Some servers may not be able to monitor the command and data connections simultaneously, in which case some special action will be necessary to get the server's attention. The exact form of the "special action" is undefined; but the following ordered format is tentatively recommended:

a. User system inserts the TELNET "Interrupt Process" (IP) signal in the TELNET stream.

b. User system sends the TELNET "Synch" signal

c. User system inserts the command (e.g., ABOR) in the TELNET stream.

d. 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.)

5.8 FTP replies. Replies to File Transfer Protocol commands are 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. The details of the command-reply sequence are made explicit in a set of state diagrams below. 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) 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.

5.8.1 FTP reply defined. 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 command 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 a Space <SP>, optionally some text, and the TELNET end-of-line code. 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 (also called spontaneous replies) which may interrupt another reply. 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.

5.8.2 Three-digit reply defined. 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.)

5.8.2.1 First digit values. There are five values for the first digit of the reply code:

5.8.2.1.1 Positive preliminary reply (1yz). 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.

5.8.2.1.2 Positive completion reply (2yz). The requested action has been successfully completed. A new request may be initiated.

5.8.2.1.3 Positive intermediate reply (3yz). 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.

5.8.2.1.4 Transient negative completion reply (4yz). 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.)

5.8.2.1.5 Permanent negative completion reply (5yz). 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.)

5.8.2.2 Second digit values. The following function groupings are encoded in the second digit:

5.8.2.2.1 Syntax (x0z). These replies refer to syntax errors, syntactically correct commands that don't fit any functional category, unimplemented or superfluous commands.

5.8.2.2.2 Information (x1z). These are replies to requests for information, such as status or help.

5.8.2.2.3 Connections (x2z). Replies referring to the command and data connections.

5.8.2.2.4 Authentication and accounting (x3z). Replies for the login process and accounting procedures.

5.8.2.2.5 Unspecified (x4z).

5.8.2.2.6 File system (x5z). These replies indicate the status of the Server file system vis-a-vis the requested transfer or other file system action.

5.8.2.3 Third digit values. 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 recommended, rather than mandatory, and may even change according to the command with which it is associated. The reply codes, on the other hand, must 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. 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 TOPS20 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 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.

5.8.3 Reply codes by function groups.

a. 200 Command okay.

b. 500 Syntax error, command unrecognized.
[This may include errors such as command line too long.]

c. 501 Syntax error in parameters or arguments.

d. 202 Command not implemented, superfluous at this site.

e. 502 Command not implemented.

f. 503 Bad sequence of commands.

g. 504 Command not implemented for that parameter.

h. 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 server's equivalent marker. (Note the spaces between markers and "=".)

i. 119 Terminal not available, will try mailbox.

j. 211 System status, or system help reply.

k. 212 Directory status.

l. 213 File status.

m. 214 Help message.
(On how to use the server or the meaning of a particular nonstandard command. This reply is useful only to the human user.)

n. 215 <scheme> is the preferred scheme.

o. 120 Service ready in nnn minutes.

p. 220 Service ready for new user.

q. 221 Service closing TELNET connection.
(logged out if appropriate)

r. 421 Service not available, closing TELNET connection.
[This may be a reply to any command if the service knows it must shut down.]

s. 125 Data connection already open; transfer starting.

t. 225 Data connection open; no transfer in progress.

u. 425 Can't open data connection.

v. 226 Closing data connection; requested file action successful.
(For example, file transfer or file abort.)

w. 426 Connection closed; transfer aborted.

x. 227 Entering Passive Mode. h1,h2,h3,h4,p1,p2.

y. 230 User logged in, proceed.

z. 530 Not logged in.

aa. 331 User name okay, need password.

bb. 332 Need account for login.

cc. 532 Need account for storing files.

dd. 150 File status okay; about to open data connection.

ee. 151 User not local; Will forward to <user>@<host>.

ff. 152 User Unknown; Mail will be forwarded by the operator.

gg. 250 Requested file action okay, completed.

hh. 350 Requested file action pending further information.

ii. 450 Requested file action not taken: file unavailable (e.g. file busy).

jj. 550 Requested action not taken: file unavailable (e.g. file not found, no access).

kk. 451 Requested action aborted: local error in processing.

ll. 551 Requested action aborted: page type unknown.

mm. 452 Requested action not taken: insufficient storage space in system

nn. 552 Requested file action aborted: exceeded storage allocation. (for current directory or dataset)

oo. 553 Requested action not taken: file name not allowed

pp. 354 Start mail input; end with <CR><LF>.<CR><LF>.

5.8.4 Numeric order list of reply codes.

a. 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 server's equivalent marker. (note the spaces between markers and "=".)

b. 119 Terminal not available, will try mailbox.

c. 120 Service ready in nnn minutes.

d. 125 Data connection already open; transfer starting.

e. 150 File status okay; about to open data connection.

f. 151 User not local; Will forward to <user>@<host>.

g. 152 User Unknown; Mail will be forwarded by the operator.

h. 200 Command okay.

i. 202 Command not implemented, superfluous at this site.

j. 211 System status, or system help reply.

k. 212 Directory status.

l. 213 File status.

m. 214 Help message.
(On how to use the server or the meaning of a particular nonstandard command. This reply is useful only to the human user.)

n. 215 <scheme> is the preferred scheme.

o. 220 Service ready for new user.

p. 221 Service closing TELNET connection.
(Logged out if appropriate.)

q. 225 Data connection open; no transfer in progress.

r. 226 Closing data connection; requested file action successful.
(For example, file transfer or file abort.)

s. 227 Entering Passive Mode. h1,h2,h3,h4,p1,p2

t. 230 User logged in, proceed.

u. 250 Requested file action okay, completed.

v. 331 User name okay, need password.

w. 332 Need account for login.

x. 350 Requested file action pending further information.

y. 354 Start mail input; end with <CR><LF>.<CR><LF>.

z. 421 Service not available, closing TELNET connection.
[This may be a reply to any command if the service knows it must shut down.]

aa. 425 Can't open data connection.

bb. 426 Connection closed; transfer aborted.

cc. 450 Requested file action not taken: file unavailable (e.g. file busy).

dd. 451 Requested action aborted: local error in processing.

ee. 452 Requested action not taken: insufficient storage space in system.

ff. 500 Syntax error, command unrecognized.
[This may include errors such as command line too long.]

gg. 501 Syntax error in parameters or arguments.

hh. 502 Command not implemented.

ii. 503 Bad sequence of commands.

jj. 504 Command not implemented for that parameter.

kk. 530 Not logged in.

ll. 532 Need account for storing files.

mm. 550 Requested action not taken: file unavailable (e.g. file not found, no access).

nn. 551 Requested action aborted: page type unknown.

oo. 552 Requested file action aborted: exceeded storage allocation (for current directory or dataset).

pp. 553 Requested action not taken: file name not allowed.

5.9 Declarative specifications.

5.9.1 Minimum implementation. In order to make FTP workable without needless error messages, the following minimum implementation is required for all servers:

TYPE - ASCII Non-print
MODE - Stream
STRUCTURE - File, Record
COMMANDS - USER, QUIT, PORT,
TYPE, MODE, STRU,
for the default values
RETR, STOR,
NOOP.

The default values for transfer parameters are:

TYPE - ASCII Non-print
MODE - Stream
STRU - File

All Hosts must accept the above as the standard defaults.

5.9.2 Connections. The server protocol interpreter shall "listen" on Port L. The user or user protocol interpreter shall initiate the full-duplex command connection. Serverand userprocesses should follow the conventions of the TELNET protocol as specified in MIL-STD 1782, TELNET Protocol. Servers are under no obligation to provide for editing of command lines and may specify that it be done in the user Host. The command connection 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 port; this may be the default user port (U) or a port specified in the PORT command. The server shall initiate the data connection from his own default data port (L-1) using the specified user data port. The direction of the transfer and the port used will be determined by the FTP service command.

5.9.2.1 Command-reply sequence. When data is to be transferred between two servers, A and B (refer to Figure 2), the user-PI, C, sets up command connections with both server-PI's. One of the servers, say A, is then sent a PASV command telling him to "listen" on his data port rather than initiate a connection when he receives a transfer service command. When the user-PI receives an acknowledgment to the PASV command, which includes the identity of the host and port being listened on, the user-PI then sends A's port, a, to B in a PORT command; a reply is returned. The user-PI may then send the corresponding service commands to A and B. Server B initiates the connection 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 : Connect	C->B : Connect
C->A : PASV	
A->C : 227 Entering Passive Mode. A1,A2,A3,A4,a1,a2	
	C->B : PORT A1,A2,A3,A4,a1,a2
	B->C : 200 Okay
C-> : STOR	C->B : RETR
B->A : Connect to HOST-A, PORT-a	

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 port BEFORE sending the transfer request). To prevent a race condition here, the server sends a reply (226) after closing the data connection (or if the connection is left open, a "file transfer completed" reply (250) and the user-PI should wait for one of these replies before issuing a new transfer command.

5.9.3 Commands. The commands are TELNET character string transmitted over the command 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, RETR, Retr, retr, ReTr, or rETr may represent the retrieve command. 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. 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.

5.9.3.1 FTP command list. The following are the FTP commands:

```
USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <account information> <CRLF>
REIN <CRLF>
QUIT <CRLF>
PORT <SP> <Host-port> <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>
CWD <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>
```

5.9.3.2 FTP command syntax. The syntax of the above argument fields (using BNF notation where applicable) is:

```
<username> ::= <string>
<password> ::= <string>
<account information> ::= <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> ::= printable characters, any
                ASCII code 33 through 126
<byte size> ::= any decimal integer 1 through 255
<Host-port> ::= <Host-number>,<Port-number>
<Host-number> ::= <number>,<number>,<number>,<number>
<Port-number> ::= <number>,<number>
<number> ::= any decimal integer 0 through 255
<ident> ::= <string>
<scheme> ::= R   T   ?
<form code> ::= N   T   C
<type code> ::= A [<SP> <form code>]
                E [<SP> <form code>]
                I
                L <SP> <byte size>
<structure code> ::= F   R   P
<mode code> ::= S   B   C
<pathname> ::= <string>
```

5.10 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. One important group of informational replies is the connection greetings. Under normal circumstances, a server will send a 220 reply, "awaiting input", when the connection 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 120 "expected delay" reply imsediately and a 220 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.

5.10.1 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 indented and 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.

```
Connection Establishment
   120
      220
   220
   421
Login
   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
```

```
Logout
   QUIT
      221
      500
   REIN
      120
          220
      220
      421
      500, 502
Transfer parameters
   PORT
      200
      500, 501, 421, 530
   PASV
      227
      500, 501, 502, 421, 530
   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, 551, 552
      532, 450, 452, 553
      500, 501, 421, 530
RETR
   125, 150
      (110)
      226, 250
   425, 426, 451
   450, 550
   500, 501, 421, 530
```

```
LIST, NLST
    125, 150
        226, 250
        425, 426, 451
    450
    500, 501, 502, 421, 530
APPE
    125, 150
        (110)
        226, 250
        425, 426, 451, 551, 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
DELE, CWD
    250
    450, 550
    500, 501, 502, 421, 530
ABOR
    225, 226
    500, 501, 502, 421
  Informational commands
    STAT
        211, 212, 213
        450
        500, 501, 502, 421, 530
    HELP
        211, 214
        500, 501, 502, 421

  Miscellaneous commands
    SITE
        200
        202
        500, 501, 530
    NOOP
        200
        500 421
```

5.11 State diagrams. Here are presented state diagrams for a very simple 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".

Figure 9 represents the largest group of FTP commands.

5.11.1 Largest group of FTP commands.

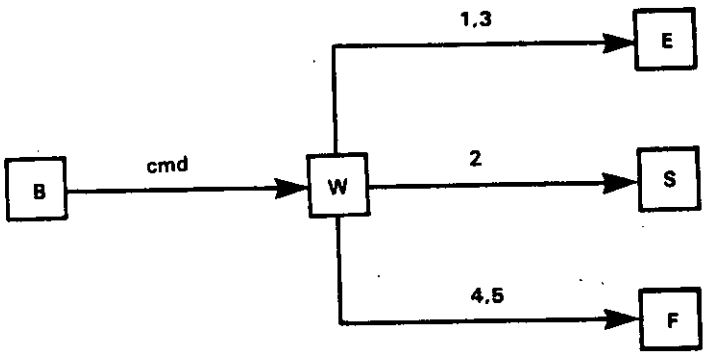


FIGURE 9. Largest group of FTP commands.

Figure 9 models the commands ABOR, ALLO, DELE, CWD, HELP, MODE, MRCP, MRCP, MRSQ, NOOP, PASV, QUIT, SITE, PORT, STAT, STRU and TYPE.

5.11.2 Other large group of FTP commands. The other large group of commands is represented by a very similar diagram.

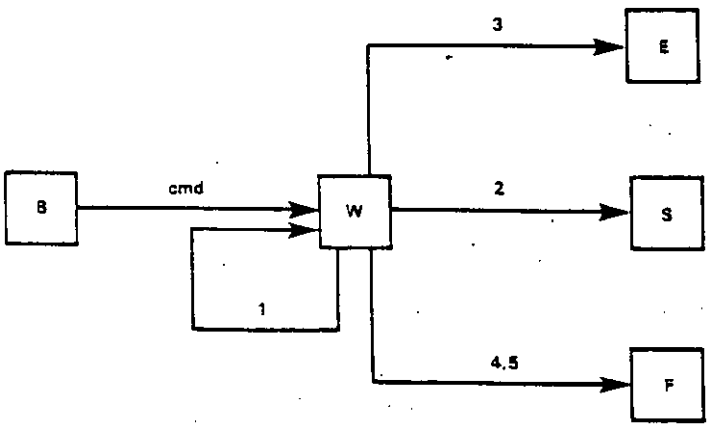


FIGURE 10. Other group of FTP commands.

Figure 10 models the commands APPE, 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.

5.11.3 Rename sequence command. The remaining diagrams model command sequences, perhaps the simplest of these is the rename sequence shown in Figure 11.

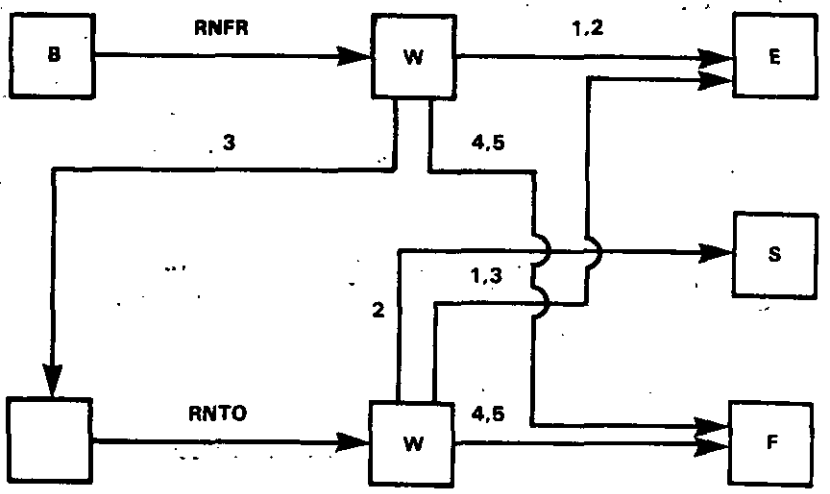


FIGURE 11. Rename sequence.

5.11.4 Restart command. Figure 12 is a simple model of the Restart command where "cmd" is APPE, STOR, or RETR Note that the Restart and Rename models are similar. The Restart differs only in the treatment of 100 series replies at the second stage.

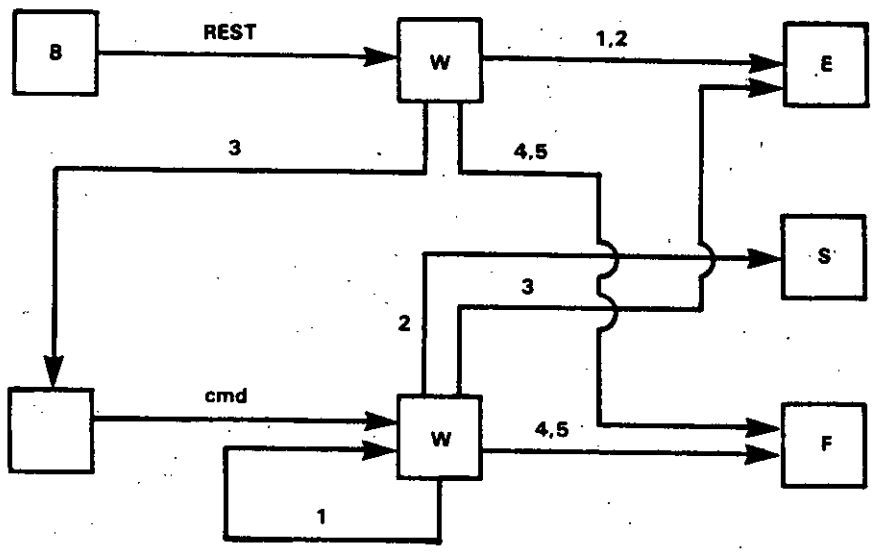


FIGURE 12. Restart command.

5.11.5 Login sequence. The most complicated diagram is for login sequence:

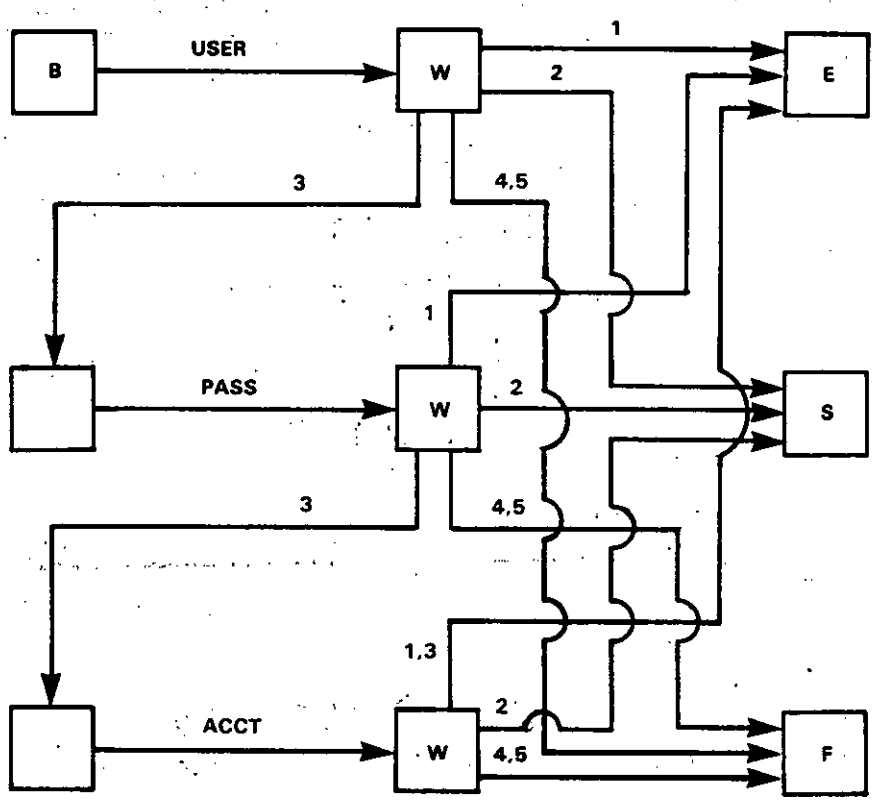


FIGURE 13. Login sequence

5.11.6 Command and reply interchange. Finally, is presented a generalized diagram that could be used to model the command and reply interchange:

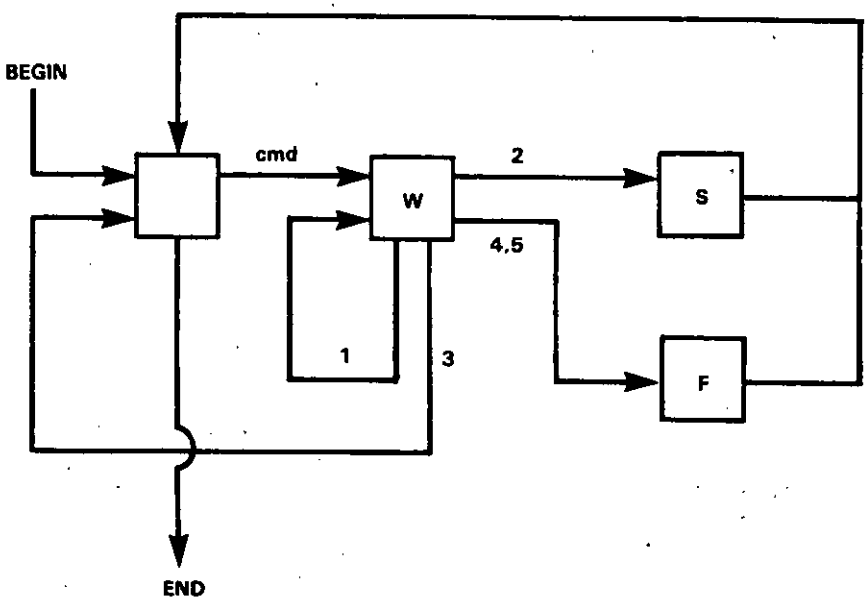


FIGURE 14. Command and reply interchange.

6. TYPICAL FTP SCENARIO

6.1 Scenario. 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) opsys<CR>	Connect to Host S, port L, establishing command connections <---- 220 Service ready <CRLF>
username Doe <CR>	USER Doe<CRLF>---->
	<---- 331 User name ok, need password<CRLF>
password pswd <CR>	PASS mumble<CRLF>---->
	<---- 230 User logged in.<CRLF>
retrieve (local type) ASCII<CR> (local pathname) test 1 <CR> (for.pathname) test.p11<CR>	User-FTP opens local file in ASCII. RETR test.p11<CRLF> ---->
	<---- 150 File status okay; about to open data connection Server makes data connection to port U
<CRLF>	
	<---- 226 Closing data connection, file transfer successful<CRLF>
type Image<CR>	TYPE I<CRLF> --
	<---- 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> ---->
	<---- 450 Access denied<CRLF>
terminate	QUIT <CRLF> ---->
	Server closes all connections.

6.2 Connection establishment. The FTP command connection is established via TCP between the user process port U and the server process port L. This protocol is assigned the service port 21 (25 octal), that is L=21.

Custodians:
Army - CR
Navy - OM
Air Force - 90

Preparing Activity:
DCA - DC

(Project IPSC-0166-02)

Review Activities:
Army - SC, CR, AD
Navy - AS, YD, MC, OM, ND, NC, EC, SA
Air Force - 1, 11, 13, 17, 90, 99

Other Interest:
NSA - NS
TRI-TAC-TT

☆ U.S. GOVERNMENT PRINTING OFFICE: 1984-505-038/A4604

INSTRUCTIONS: In a continuing effort to make our standardization documents better, the DoD provides this form for use in submitting comments and suggestions for improvements. All users of military standardization documents are invited to provide suggestions. This form may be detached, folded along the lines indicated, taped along the loose edge *(DO NOT STAPLE)*, and mailed. In block 5, be as specific as possible about particular problem areas such as wording which required interpretation, was too rigid, restrictive, loose, ambiguous, or was incompatible, and give proposed wording changes which would alleviate the problems. Enter in block 6 any remarks not related to a specific paragraph of the document. If block 7 is filled out, an acknowledgement will be mailed to you within 30 days to let you know that your comments were received and are being considered.

NOTE: This form may not be used to request copies of documents, nor to request waivers, deviations, or clarification of specification requirements on current contracts. Comments submitted on this form do not constitute or imply authorization to waive any portion of the referenced document*(s)* or to amend contractual requirements.

(Fold along this line)

(Fold along this line)

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 4966 Alexandria, VA

POSTAGE WILL BE PAID BY

DIRECTOR
DEFENSE COMMUNICATIONS AGENCY
ATTN: CODE R130
1860 WIEHLE AVENUE
RESTON, VA 22090-5500

STANDARDIZATION DOCUMENT IMPROVEMENT PROPOSAL

(See Instructions – Reverse Side)

1. DOCUMENT NUMBER	2. DOCUMENT TITLE
MIL STD 1780	FILE TRANSER PROTOCOL

3a. NAME OF SUBMITTING ORGANIZATION

b. ADDRESS *(Street, City, State, ZIP Code)*

4. TYPE OF ORGANIZATION *(Mark one)*

- ☐ VENDOR
- ☐ USER
- ☐ MANUFACTURER
- ☐ OTHER *(Specify)*: _____

5. PROBLEM AREAS

a. Paragraph Number and Wording:

b. Recommended Wording:

c. Reason/Rationale for Recommendation:

6. REMARKS

7a. NAME OF SUBMITTER *(Last, First, MI)* – Optional	b. WORK TELEPHONE NUMBER *(Include Area Code)* – Optional
c. MAILING ADDRESS *(Street, City, State, ZIP Code)* – Optional	8. DATE OF SUBMISSION *(YYMMDD)*

DD FORM 82 MAR 1426 PREVIOUS EDITION IS OBSOLETE.