File Transfer Protocol
(Aug. 12, 1973)
RFC 542 NIC 17759

Nancy J. Neigus
Bolt Beranek and Newman, Inc.
Cambridge, Mass.

See Also: RFCs 354, 454, 495

File Transfer Protocol for the ARPA Network

PREFACE

This document is the result of several months discussion via RFC (relevant numbers are 430, 448, 454, 463, 468, 478, 480), followed by a meeting of the FTP committee at BBN on March 16, followed by further communication among committee members. There are a considerable number of changes for the last "official" version, see RFCs 354, 385, but the gross structure remains the same. The places to look for differences are (1) in the definitions pf types and modes, (2) in the specification of the data connection and data sockets, (3) in the command-reply sequences, (4) in the functions dependent on the TELNET protocol (FTP has been altered to correspond to the new TELNET spec). The model has been clarified and enlarged to allow inter-server file transfer, and several new commands have been added to accommodate more specialized (or site-specific) functions. It is my belief that this new specificiation reflects the views expressed by the committee at the above-mentioned meeting and in subsequent conversations.

The large number of incompatibilities would complicate a phased implementation schedule, such as is in effect for the TELNET protocol. Therefore we have assigned a new socket, decimal 21, as a temporary logger socket for the new version and a change-over date of 1 February 1974. Until that date the old (354, 385) version of FTP will be available on Socket 3 and the new version (attached) should be implemented on Socket 21. On 1 February the new version will shift to Socket 3 and the old disappear from view.

The File Transfer protocol should be considered stable at least until February, though one should feel free to propose further changes via RFC. (Implementation of new commands on an experimental basis is encouraged and should also be reported by RFC.) In addition, members of the FTP committee may be contacted directly about changes. Based on attendance at the March 16 meeting, they are:

Abhay Bhushan MIT-DMCG
Bob Braden UCLA-CCN
Bob Bressler BBN-NET
Bob Clements BBN-TENEX
John Day ILL-ANTS
Peter Deutsch PARC-MAXC
Wayne Hathaway AMES-67
Mike Kudlick SRI-ARC
Alex McKenzie BBN-NET
Bob Merryman UCSD-CC
Nancy Neigus BBN-NET
Mike Padlipsky MIT-Multics
Jim Pepin USC-44
Ken Pogran MIT-Multics
Jon Postel UCLA-NMC

Milton Reese FNWC
Brad Reussow HARV-10
Marc Seriff MIT-DMCG
Ed Taft HARV-10
Bob Thomas BBN-TENEX
Ric Werme CMU-10
Jim White SRI-ARC

I would especially like to thank Bob Braden, Ken Pogran, Wayne Hathaway, Jon Postel, Ed Taft and Alex McKenzie for their help in preparing this document.

NJN/jm

FILE TRANSFER PROTOCOL

INTRODUCTION

The File Transfer Protocol (FTP) is a protocol for file transfer between Hosts (including Terminal Interface Message Processors (TIPs)) on the ARPA Computer Network (ARPANET). The primary function of FTP is to transfer files efficiently and reliably among Hosts and to allow the convenient use of remote file storage capabilities.

The objectives of FTP are 1) to promote sharing of files (computer programs and/or data), 2) to encourage indirect or implicit (via programs) use of remote computers, 3) to shield a user from variations in file storage systems among Hosts, and 4) to transfer data reliably and efficiently. FTP, though usable directly by a user at a terminal, is designed mainly for use by programs.

The attempt in this specification is to satisfy the diverse needs of users of maxi-Hosts, mini-Hosts, TIPs, and the Datacomputer, with a simple, and easily implemented protocol design.

This paper assumes knowledge of the following protocols described in NIC #7104:

The Host-Host Protocol

The Initial Connection Protocol

The TELNET Protocol

DISCUSSION

In this section, the terminology and the FTP model are discussed. The terms defined in this section are only those that have special significance in FTP.

TERMINOLOGY

ASCII

The USASCII character set as defined in NIC #7104. In FTP, ASCII characters are defined to be the lower half of an eight-bit code set (i.e., the most significant bit is zero).

access controls

Access controls define users' access privileges to the use of a system, and to the files in that system. Access controls are necessary to prevent unauthorized or accidental use of files.

It is the prerogative of a server-FTP process to provide access controls.

byte size

The byte size specified for the transfer of data. The data connection is opened with this byte size. The data connection byte size is not necessarily the byte size in which data is to be stored in a system, nor the logical byte size for interpretation of the structure of the data.

data connection

A simplex connection over which data is transferred, in a specified byte size, mode and type. The data transferred may be a part of a file, an entire file or a number of files. The path may be between a server-DTP and a user-DTP, or between two server-DTPs.

data socket

The passive data transfer process "listens" on the data socket for an RFC from the active transfer process (server) in order to open the data connection. The server has fixed data sockets; the passive process may or may not.

EOF

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

EOR

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

error recovery

A procedure that allows a user to recover from certain errors such as failure of either Host system or transfer process. In FTP, error recovery may involve restarting a file transfer at a given checkpoint.

FTP commands

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

file

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

mode

The mode in which data is to be transferred via the data connection. The mode defines the data format during transfer including EOR and EOF. The transfer modes defined in FTP are described in the Section on Transmission Modes.

NVT

The Network Virtual Terminal as defined in the ARPANET TELNET Protocol.

NVFS

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

pathname

Pathname is defined to be the character string which must be input to a file system by a user in order to identify a file. Pathname normally contains device and/or directory names, and file name specification. FTP does not yet specify a standard pathname convention. Each user must follow the file naming conventions of the file systems he wishes to use.

record

A sequential file may be structured as a number of contiguous parts called records. Record structures are supported by FTP but a file need not have record structure.

reply

A reply is an acknowledgment (positive or negative) sent from server to user via the TELNET connections in response to FTP commands. The general form of a reply is a completion code (including error codes) followed by a text string. The codes are for use by programs and the text is usually intended for human users.

server-DTP

The data transfer process, in its normal "active" state, establishes the data connection by RFC to the "listening" data socket, sets up parameters for transfer and storage, and tranfers data on command from its PI. The DTP can be placed in a "passive" state to listen for, rather than initiate, an RFC on the data socket.

server-FTP process

A process or set of processes which perform the function of file transfer in cooperation with a user-FTP process and, possibly, another server. The functions consist of a protocol interpreter (PI) and a data transfer process (DTP).

server-PI

The protocol interpreter "listens" on Socket 3 for an ICP from a user-PI and establishes a TELNET communication connection. It receives standard FTP commands from the user-PI, sends replies, and governs the server-DTP.

TELNET connections

The full-duplex communication path between a user-PI and a server-PI. The TELNET connections are established via the standard ARPANET Initial Connection Protocol (ICP).

type

The data representation type used for data transfer and storage. Type implies certain transformations between the time of data storage and data transfer. The representation types defined in FTP are described in the Section on Establishing Data Connections.

user

A human being or a process on behalf of a human being wishing to obtain file transfer service. The human user may interact directly with a server-FTP process, but use of a user-FTP process is preferred since the protocol design is weighted towards automata.

user-DTP

The data transfer process "listens" on the data socket for an RFC from a server-FTP process. If two servers are transferring data between them, the user-DTP is inactive.

user-FTP process

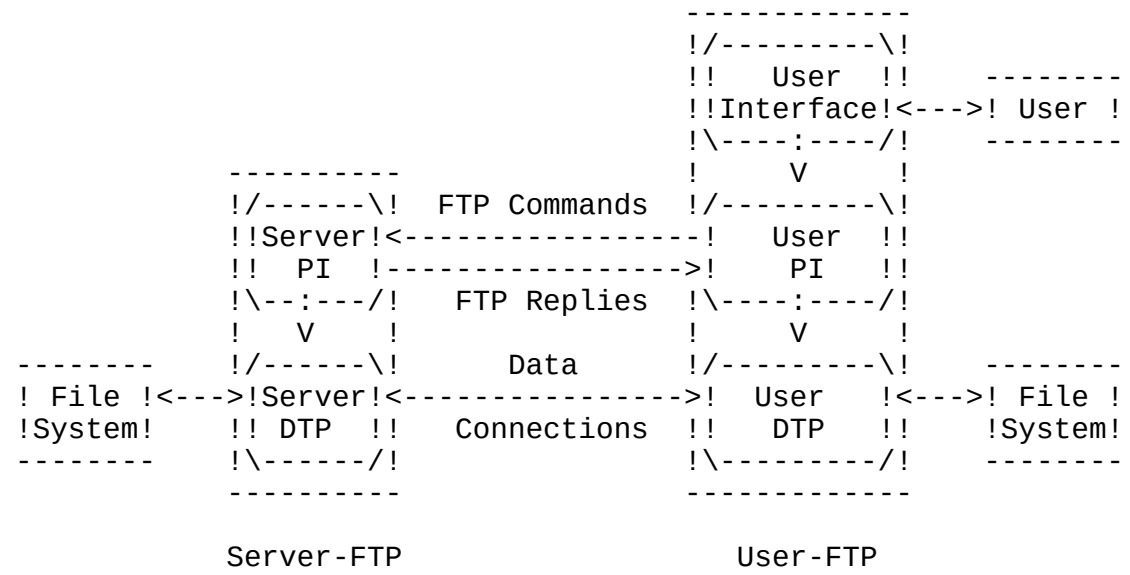
A set of functions including a protocol interpreter, a data transfer process and a user interface which together perform the function of file transfer in cooperation with one or more server-FTP processes. The user interface allows a local language to be used in the command-reply dialogue with the user.

user-PI

The protocol interpreter initiates the ICP to the server-FTP process, initiates FTP commands, and governs the user-DTP if that process is part of the file transfer.

THE FTP MODEL

With the above definitions in mind, the following model (shown in Figure 1) may be diagrammed for an FTP service.



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

Figure 1 Model for FTP Use

In the model described in Figure 1, the user-protocol interpreter initiates the TELNET connections. At the initiation of the user, standard FTP commands are generated by the user-PI and transmitted to the server process via the TELNET connections. (The user may establish a direct TELNET connection to the server-FTP, from a TIP terminal for example, and generate standard FTP commands himself, by-passing the user-FTP process.) Standard replies are sent from the server-PI to the user-PI over the TELNET connections in response to the commands.

The FTP commands specify the parameters for the data connection (data socket, byte size, transfer mode, representation type, and structure) and the nature of file system operation (store, retrieve, append, delete, etc.). The user-DTP or its designate should "listen" on the specified data socket, and the server initiate the data connection and data transfer in accordance with the specified parameters. It should be noted that the data socket need not be in the same Host that initiates the FTP commands via the TELNET connections, but the user or his user-FTP process must ensure a "listen" on the specified data socket. It should also be noted that two data connections, one for send and the other for receive, may exist simultaneously.

In another situation a user might wish to transfer files between two Hosts, neither of which is his local Host. He sets up TELNET connections to the two servers and then arranges for a data connection between them. In this manner control information is passed to the user-PI but data is transferred between he server data transfer processes. Following is a model of this server-server interaction.

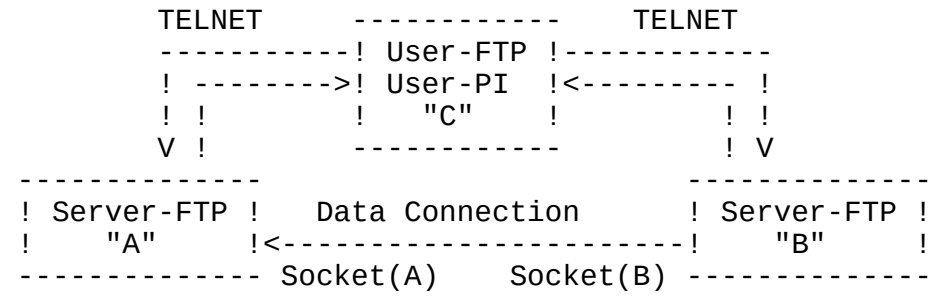


Figure 2

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

DATA TRANSFER FUNCTIONS

Files are transferred only via the data connection(s). The TELNET connection is used for the transfer of commands, which describe the functions to be performed, and the replies to these commands (see the Section on FTP Replies). Several commands are concerned with the transfer of data between Hosts. These data transfer commands include the BYTE, MODE, and SOCKet commands which specify how the bits of the data are to be transmitted, and the STRUcture and TYPE commands, which are used to define the way in which the data are to be represented. The transmission and representation are basically independent but "Stream" transmission mode is dependent on the file structure attribute and if "Compressed" transmission mode is used the nature of the filler byte depends on the representation type.

DATA REPRESENTATION AND STORAGE

Data is transferred from a storage device in the sending Host to a storage device in the receiving Host. Often it is necessary to perform certain transformations on the data because data storage representations in the two systems are different. For example, NVT-ASCII has different data storage representations in diffeent systems. PDP-10's generally store NVT-ASCII as five 7-bit ASCII characters, left-justified in a 36-bit word. 360's store NVT-ASCII as 8-bit EBCDIC codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit word. It may be desirable to convert characters into the standard NVT-ASCII representation when transmitting text between dissimilar systems. The sending and receiving sites would have to perform the necessary transformations between the standard representation and their internal representations.

A different problem in representation arises when transmitting binary data (not character codes) between Host systems with different word lengths. It is not always clear how the sender should send data, and the receiver store it. For example, when transmitting 32-bit bytes from a 32-bit word-length system to a 36-bit word-length system, it may be desirable (for reasons of efficiency and usefulness) to store the 32-bit bytes right-justified in a 36-bit word in the latter system. In any case, the user should have the option of specifying data representation and transformation functions. It should be noted that FTP provides for very limited data type representations. Transformations desired beyond this limited capability should be

performed by the user directly or via the use of the Data Reconfiguration Sevice (DRS, RFC #138, NIC #6715). Additonal representation types may be defined later if there is a demonstrable need.

Data representations are handled in FTP by a user specifying a representation type. This type may implicitly (as in ASCII or EBCDIC) or explicitly (as in Local byte) define a byte size for interpretation which is referred to as the "logical byte size." This has nothing to do with the byte size used for transmission over the data connection(s) (called the "transfer byte size") and the two should not be confused. For example, NVT-ASCII has a logical byte size of 8 bits but an ASCII file might be transferred using a transfer byte size of 32. If the type is Local byte, then the TYPE command has an obligatory second parameter specifying the logical byte size.

The types ASCII and EBCDIC also take a second (optional) parameter; this is to indicate what kind of vertical format control, if any, is associated with a file. The following data representation types are defined in FTP:

ASCII Format

This is the default type and must be accepted by all FTP implementations. It is intended primarily for the transfer of text files, except when both Hosts would find the EBCDIC type more convenient.

The sender converts the data from his internal character representation to the standard 8-bit NVT-ASCII representation (see the TELNET specification). The receiver will convert the data from the standard form to his own internal form.

In accordance with the NVT standard, the <CRLF> sequence should be used, where necessary, to denote the end of a line of text. (See the discussion of file structure at the end of the Section on Data Representation and Storage).

Using the standard NVT-ASCII representation means that data must be interpreted as 8-bit bytes. If the BYTE command (see the Section on Transfer Parameter Commands) specifies a transfer byte size different from 8 bits, the 8-bit ASCII characters should be packed contiguously without regard for transfer byte boundaries.

The Format parameter for ASCII and EBCDIC types is discussed below.

EBCDIC Format

This type is intended for efficient transfer between Hosts which use EBCDIC for their internal character representation.

For transmission the data are represented as 8-bit EBCDIC characters. The character code is the only difference between the functional specifications of EBCDIC and ASCII types.

End-of-line (as opposed to end-of-record--see the discussion of structure) will probably be rarely used with EBCDIC type for purposes of denoting structure, but where it is necessary the <NL> character should be used.

A character file may be transferred to a Host for one of three purposes: for printing, for storage and later retrieval, or for processing. If a file is sent for printing, the receiving Host must know how the vertical format control is represented. In the second case, it must be possible to store a file at a Host and then retrieve it later in exactly the same form. Finally, it ought to be possible to move a file from one Host to another and process the file at the second Host without undue trouble. A single ASCII or EBCDIC format does not satisfy all these conditions and so these types have a second parameter specifying one of the following three formats:

Non-print

This is the default format to be used if the second (format) parameter is omitted. Non-print format must be accepted by all FTP implementations.

The file need contain no vertical format information. If it is passed to a printer process, this process may assume standard values for spacing and margins.

Normally, this format will be used with files destined for processing or just storage.

TELNET Format Controls

The file contains ASCII/EBCDIC vertical format controls (i.e., <CR>, <LF>, <NL>, <VT>, <FF>) which the printer process will interpret appropriately. <CRLF>, in exactly this sequence, also denotes end-of-line.

Carriage Control (ASA)

The file contains ASA (FORTRAN) vertical format control characters. (See NWG/RFC #189 Appendix C and Communications of

the ACM, Vol. 7, No. 10, 606 (Oct. 1964)). In a line or a record, formatted according to the ASA Standard, the first character is not to be printed. Instead it should be used to determine the vertical movement of the paper which should take place before the rest of the record is printed. The ASA Standard specifies the following control characters:

Character	Vertical Spacing
blank	Move paper up one line
0	Move paper up two lines
1	Move paper to top of next page
+	No movement, i.e., overprint

Clearly there must be some way for a printer process to distinguish the end of the structural entity. If a file has record structure (see below) this is no problem; records will be explicitly marked during transfer and storage. If the file has no record structure, the <CRLF> end-of-line sequence is used to separate printing lines, but these format effectors are overridden by the ASA controls.

Image

The data are sent as contiguous bits which, for transfer, are packed into transfer bytes of the size specified in the BYTE command. The receiving site must store the data as contiguous bits. The structure of the storage system might necessitate the padding of the file (or of each record, for a record-structured file) to some convenient boundary (byte, word or block). This padding, which must be all zeroes, may occur only at the end of the file (or at the end of each record) and there must be a way of identifying the padding bits so that they may be stripped off if the file is retrieved. The padding transformation should be well publicized to enable a user to process a file at the storage site.

Image type is intended for the efficient storage and retrieval of files and for the transfer of binary data. It is recommended that this type be accepted by all FTP implementations.

Local byte Byte size

The data is transferred in logical bytes of the size specified by the obligatory second parameter, Byte size. The value of Byte size must be a decimal integer; there is no default value. The logical byte size is not necessarily the same as the transfer byte size. If there is a difference in byte sizes,

then the logical bytes should be packed contiguously, disregarding transfer byte boundaries and with any necessary padding at the end.

When the data reaches the receiving Host it will be transformed in a manner dependent on the logical byte size and the particular Host. This transformation must be invertible (that is an identical file can be retrieved if the same parameters are used) and should be well publicized by the FTP implementors.

This type is intended for the transfer of structured data. For example, a user sending 36-bit floating-point numbers to a Host with a 32-bit word could send his data as Local byte with a logical byte size of 36. The receiving Host would then be expected to store the logical bytes so that they could be easily manipulated; in this example putting the 36-bit logical bytes into 64-bit double words should suffice.

A note of caution about parameters: a file must be stored and retrieved with the same parameters if the retrieved version is to be identical to the version originally transmitted. Conversely, FTP implementations must return a file identical to the original if the parameters used to store and retrieve a file are the same.

In addition to different representation types, FTP allows the structure of a file to be specified. Currently two file structures are recognized in FTP: file-structure, where there is no internal structure, and record-structure, where the file is made up of records. File-structure is the default, to be assumed if the STRUcture command has not been used but both structures must be accepted for "text" files (i.e., files with TYPE ASCII or EBCDIC) by all FTP implementations. The structure of a file will affect both the transfer mode of a file (see the Section on Transmission Modes) and the interpretation and storage of the file.

The "natural" structure of a file will depend on which Host stores the file. A source-code file will usually be stored on an IBM 360 in fixed length records but on a PDP-10 as a stream of characters partitioned into lines, for example by <CRLF>. If the transfer of files between such disparate sites is to be useful, there must be some way for one site to recognize the other’s assumptions about the file.

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 retreieved using record structure.

In the case of a file being sent with file-structure to a record-oriented Host, there exists the question of what criteria the Host should use to divide the file into records which can be processed locally. If this division is necessary the FTP implementation should use the end-of-line sequence, <CRLF> for ASCII, or <NL> for EBCDIC text files, as the delimiter. If an FTP implementation adopts this technique, it must be prepared to reverse the transformation if the file is retrieved with file-structure.

ESTABLISHING DATA CONNECTIONS

The mechanics of transferring data consists of setting up the data connection to the appropriate sockets and choosing the parameters for transfer--byte size and mode. Both the user and the server-DTPs have default data sockets; these are the two sockets (for send and receive) immediately following the standard ICP TELNET socket ,i.e., (U+4) and (U+5) for the user-process and (S+2), (S+3) for the server. The use of default sockets will ensure the security of the data transfer, without requiring the socket information to be explicitly exchanged.

The byte size for the data connection is specified by the BYTE command, or, if left unspecified, defaults to 8-bit bytes. This byte size is relevant only for the actual transfer of the data; it has no bearing on representation of the data within a Host’s file system. The protocol does not require servers to accept all possible byte sizes. Since the use of various byte sizes is intended for efficiency of transfer, servers may implement only those sizes for which their data transfer is efficient including the default byte size of 8 bits.

The passive data transfer process (this may be a user-DTP or a second server-DTP) shall "listen" on the data socket prior to sending a transfer request command. The FTP request command determines the direction of the data transfer and thus which data socket (odd or even) is to be used in establishing the connection. The server, upon receiving the transfer request, will initiate the data connection by RFC to the appropriate socket using the specified (or default) byte size. When the connection is opened, the data transfer begins between DTP’s, and the server-PI sends a confirming reply to the user-PI.

It is possible for the user to specify an alternate data socket by use of the SOCK command. He might want a file dumped on a TIP line printer or retrieved from a third party Host. In the latter case the user-PI sets up TELNET connections with both server-PI’s and sends

each a SOCK command indicating the fixed data sockets of the other. One server is then told (by an FTP command) to "listen" for an RFC which the other will initiate and finally both are sent the appropriate transfer commands. The exact sequence of commands and replies sent between the user-controller and the servers is defined in the Section on FTP Replies.

In general it is the server's responsibility to maintain the data connection--to initiate the RFC's and the closes. The exception to this is when the user-DTP is sending the data in a transfer mode that requires the connection to be closed to indicate EOF. The server MUST close the data connection under the following conditions:

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

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

3. The socket or byte size specification is changed by a command from the user.

4. The TELNET connections are closed legally or otherwise.

5. An irrecoverable error condition occurs.

Otherwise the close is a server option, the exercise of which he must indicate to the user-process by an appropriate reply.

TRANSMISSION MODES

The next consideration in transferring data is choosing the appropriate transmission mode. There are three modes: one which formats the data and allows for restart procedures; one which also compresses the data for efficient transfer; and one which passes the data with little or no processing. In this last case the mode interacts with the structure attribute to determine the type of processing. In the compressed mode the representation type determines the filler byte.

All data transfers must be completed with an end-of-file (EOF) which may be explicitly stated or implied by the closing of the data connection. For files with record structure, all the end-of-record markers (EOR) are explicit, including the final one.

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

The following transmission modes are defined in FTP:

Stream

The data is transmitted as a stream of bytes. There is no restriction on the representation type used; record structures are allowed, in which case the transfer byte size must be at least 3 bits!

In a record structured file EOR and EOF will each be indicated by a two-byte control code of whatever byte size is used for the transfer. The first byte of the control code will be all ones, the escape character. The second byte will have the low order bit on and zeroes elsewhere for EOR and the second low order bit on for EOF; that is, the byte will have value 1 for EOR and value 2 for EOF. EOR and EOF may be indicated together on the last byte transmitted by turning both low order bits on, i.e., the value 3. If a byte of all ones was intended to be sent as data, it should be repeated in the second byte of the control code.

If the file does not have record structure, the EOF is indicated by the sending Host closing the data connection and all bytes are data bytes.

For the purpose of standardized transfer, the sending Host will translate his internal end of line or end of record denotation into the representation prescribed by the transfer mode and file structure, and the receiving Host will perform the inverse translation to his internal denotation. An IBM 360 record count field may not be recognized at another Host, so the end of record information may be transferred as a two byte control code in Stream mode or as a flagged bit in a Block or Compressed mode descriptor. End of line in an ASCII or EBCDIC file with no record structure should be indicated by <CRLF> or <NL>, respectively. Since these transformations imply extra work for some systems, identical systems transferring non-record structured text files might wish to use a binary representation and stream mode for the transfer.

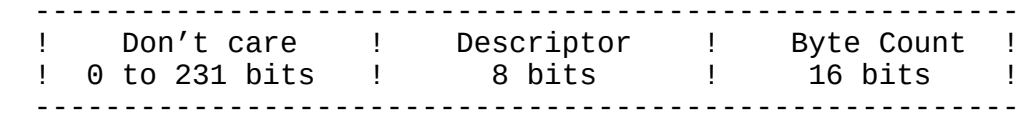
Block

The file is transmitted as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field, and descriptor code. The count field indicates the total length of the data block in bytes, thus marking the beginning of the next data block (there are no filler bits). The descriptor code defines: last block in the file (EOF) last block in the record (EOR), restart marker (see the Section on Error Recovery and Restart) or suspect data (i.e., the data

being transferred is suspected of errors and is not reliable). This last code is NOT intended for error control within FTP. It is motivated by the desire of sites exchanging certain types of data (e.g., seismic or weather data) to send and receive all the data despite local errors (such as "magnetic tape read errors"), but to indicate in the transmission that certain portions are suspect). Record structures are allowed in this mode, and any representation type may be used. There is no restriction on the transfer byte size.

The header consists of the smallest integral number of bytes whose length is greater than or equal to 24 bits. Only the LEAST significant 24 bits (right-justified) of header shall have information; the remaining most significant bits are "don't care" bits. Of the 24 bits of header information, the 16 low order bits shall represent byte count, and the 8 high order bits shall represent descriptor codes as shown below.

Integral number of bytes greater than or equal to 24 bits



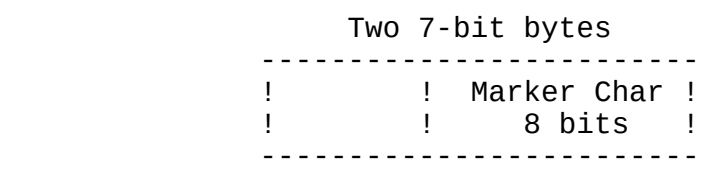
The descriptor codes are indicated by bit flags in the descriptor byte. Four codes have been assigned, where each code number is the decimal value of the corresponding bit in the byte.

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

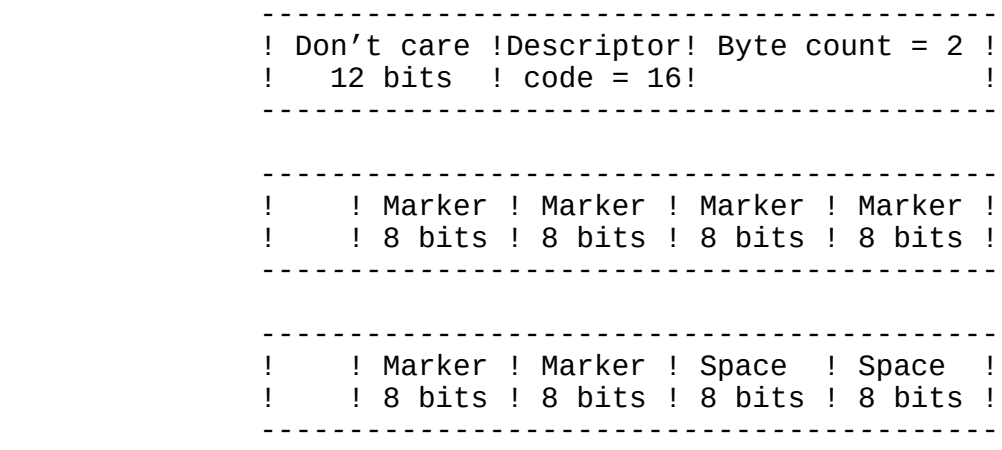
With this encoding more than one descriptor coded condition may exist for a particular block. As many bits as necessary may be flagged.

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). These marker bytes are right-justified in the smallest integral number of transfer bytes greater than or equal to 8 bits. For example, if the

byte size is 7 bits, the restart marker byte would be one byte right-justified per two 7-bit bytes as shown below:

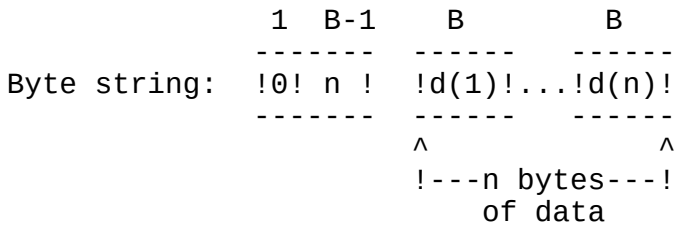


If the transfer byte size is 16 or more bits, the maximum possible number of complete marker bytes should be packed, right-justified, into each transfer byte. The restart marker should begin in the first marker byte. If there are any unused marker bytes, these should be filled with the character <SP> (Space, in the appropriate language). <SP> must not be used WITHIN a restart marker. For example, to transmit a six-character marker with a 36-bit transfer byte size, the following three 36-bit bytes would be sent:



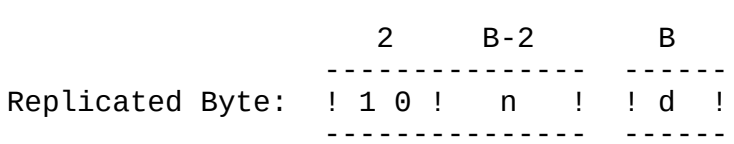
Compressed

The file is transmitted as series of bytes of the size specified by the BYTE command. There are three kinds of information to be sent: regular data, sent in a byte string; compressed data, consisting of replications or filler; and control information, sent in a two-byte escape sequence. If the byte size is B bits and n>0 bytes of regular data are sent, these n bytes are preceded by a byte with the left-most bit set to 0 and the right-most B-1 bits containing the number n.

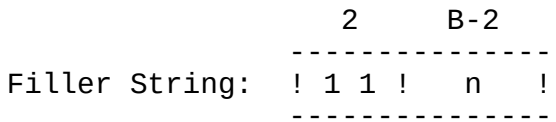


String of n data bytes d(1),..., d(n)
Count n must be positive

To compress a string of n replications of the data byte d, the following 2 bytes are sent:



A string of n filler bytes can be compressed into a single byte, where the filler byte varies with the representation type. If the type is ASCII or EBCDIC the filler byte is <SP> (Space, ASCII code 32., EBCDIC code 64). If the transfer byte size is not 8, the expanded byte string should be filled with 8-bit <SP> characters in the manner described in the definition of ASCII representation type (see the Section on Data Representation and Storage). If the type is Image or Local byte the filler is a zero byte.



The escape sequence is a double byte, the first of which is the escape byte (all zeroes) and the second of which contains descriptor codes as defined in Block mode. This implies that the byte size must be at least 8 bits, which is not much of a restriction for efficiency in this mode. The descriptor codes have the same meaning as in Block mode and apply to the succeeding string of bytes.

Compressed mode is useful for obtaining increased bandwidth on very large network transmissions at a little extra CPU cost. It is most efficient when the byte size chosen is that of the word size of the transmitting Host, and can be most effectively used to reduce the size of printer files such as those generated by RJE Hosts.

ERROR RECOVERY AND RESTART

There is no provision for detecting bits lost or scrambled in data transfer. This issue is perhaps handled best at the NCP level where it benefits most users. However, a restart procedure is provided to protect users from gross system failures (including failures of a Host, an FTP-process, or the IMP subnet).

The restart procedure is defined only for the block and compressed modes of data transfer. It requires the sender of data to insert a special marker code in the data stream with some marker information. The marker information has meaning only to the sender, but must consist of printable characters in the default or negotiated language of the TELNET connection. The marker could represent a bit-count, a record-count, or any other information by which a system may identify a data checkpoint. The receiver of data, if it implements the restart procedure, would then mark the corresponding position of this marker in the recieving system, and return this information to the user.

In the event of a system failure, the user can restart the data transfer by identifying the marker point with the FTP restart procedure. The following example illustrates the use of the restart procedure.

The sender of the data inserts an appropriate marker block in the data stream at a convenient point. The receiving Host marks the corresponding data point in its file system and conveys the last known sender and receiver marker information to the user, either directly or over the TELNET connection in a 251 reply (depending on who is the sender). In the event of a system failure, the user or controller process restarts the server at the last server marker by sending a restart command with server's marker code as its argument. The restrart command is transmitted over the TELNET connection and is immediately followed by the command (such as RETR, STOR or LIST) which was being executed when the system failure occurred.

FILE TRANSFER FUNCTIONS

The communication channel from the user-PI to the server-PI is established by ICP from the user to a standard server socket. The user protocol interpreter is responsible for sending FTP commands and interpreting the replies received; the server-PI interprets commands, sends replies and directs its DTP to set up the data connection and transfer the data. If the second party to the data transfer (the passive transfer process) is the user-DTP then it is governed through the internal protocol of the user-FTP Host; if it is a second server-DTP then it is governed by its PI on command from the user-PI.

FTP COMMANDS

The File Transfer Protocol follows the specifications of the TELNET protocol for all communications over the TELNET connection - see NIC #7104. Since, in the future, the language used for TELNET communication may be a negotiated option, all references in the next two sections will be to the "TELNET language" and the corresponding "TELNET end of line code". Currently one may take these to mean NVT-ASCII and <CRLF>. No other specifications of the TELNET protocol will be cited.

FTP commands are "TELNET strings" terminated by the "TELNET end of line code". The command codes themselves are alphabetic characters terminated by the character <SP> (Space) if parameters follow and TELNET-EOL otherwise. The command codes and the semantics of commands are described in this section; the detailed syntax of commands is specified in the Section on Commands, the reply sequences are discussed in the Section on Sequencing of Commands and Replies, and scenarios illustrating the use of commands are provided in the Section on Typical FTP Scenarios.

FTP commands may be partitioned as those specifying access-control identifiers, data transfer parameters, or FTP service requests. Certain commands (such as ABOR, STAT, BYE) may be sent over the TELNET connections while a data transfer is in progress. Some servers may not be able to monitor the TELNET and data connections simultaneously, in which case some special action will be necessary to get the server’s attention. The exact form of the "special action" is related to decisions currently under review by the TELNET committee; but the following ordered format is tentatively recommended:

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

2. User system sends the TELNET "Synch" signal

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

4. Server PI,, after receiving "IP", scans the TELNET stream for EXACTLY ONE FTP command.

(For other servers this may not be necessary but the actions listed above should have no unusual effect.)

ACCESS CONTROL COMMANDS

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

USER NAME (USER)

The argument field is a TELNET string identifying the user. The user identification is that which is required by the server for access to its file system. This command will normally be the first command transmitted by the user after the TELNET connections are made (some servers may require this). Additional identification information in the form of a password and/or an account command may also be required by some servers. Servers may allow a new USER command to be entered at any point in order to change the access control and/or accounting information. This has the effect of flushing any user, password, and account information already supplied and beginning the login sequence again. All transfer parameters are unchanged and any file transfer in progress is completed under the old acccount.

PASSWORD (PASS)

The argument field is a TELNET string identifying the user's password. This command must be immediately preceded by the user name command, and, for some sites, completes the user's identification for access control. Since password information is quite sensitive, it is desirable in general to "mask" it or suppress typeout. It appears that the server has no foolproof way to achieve this. It is therefore the responsibility of the user-FTP process to hide the sensitive password information.

ACCOUNT (ACCT)

The argument field is a TELNET string identifying the user's account. The command is not necessarily related to the USER command, as some sites may require an account for login and others only for specific access, such as storing files. In the latter case the command may arrive at any time. There are two reply codes to differentiate these cases for the automaton: when account information is required for login, the response to a successful PASSword command is reply code 331; then if a command other than ACCounT is sent, the server may remember it and return a 331 reply, prepared to act on the command after the account information is received; or he may flush the command and return a 433 reply asking for the account. On the other hand, if account information is NOT required for login, the reply to a successful PASSword command is 230; and if the

information is needed for a command issued later in the dialogue, the server should return a 331 or 433 reply depending on whether he stores (pending receipt of the ACCounT command) or discards the command, respectively.

REINITIALIZE (REIN)

This command terminates a USER, flushing all I/O and account information, except to allow any transfer in progress to be completed. All parameters are reset to the default settings and the TELNET connection is left open. This is identical to the state in which a user finds himself immediately after the ICP is completed and the TELNET connections are opened. A USER command may be expected to follow.

LOGOUT (BYE)

This command terminates a USER and if file transfer is not in progress, the server closes the TELNET connection. If file transfer is in progress, the connection will remain open for result response and the server will then close it. If the user-process is transferring files for several USERs but does not wish to close and then reopen connections for each, then the REIN command should be used instead of BYE.

An unexpected close on the TELNET connection will cause the server to take the effective action of an abort (ABOR) and a logout (BYE).

TRANSFER PARAMETER COMMANDS

All data transfer parameters have default values, and the commands specifying data transfer parameters are required only if the default parameter values are to be changed. The default value is the last specified value, or if no value has been specified, the standard default value as stated here. This implies that the server must "remember" the applicable default values. The commands may be in any order except that they must precede the FTP service request. The following commands specify data transfer parameters.

BYTE SIZE (BYTE)

The argument is a decimal integer (1 through 255) specifying the byte size for the data connection. The default byte size is 8 bits. A server may reject certain byte sizes that he has not implemented.

DATA SOCKET (SOCK)

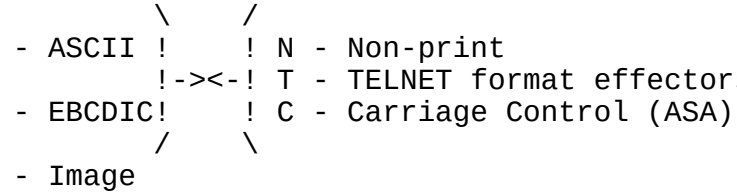
The argument is a HOST-SOCKET specification for the data socket to be used in data connection. There may be two data sockets, one for transfer from the "active" DTP to the "passive" DTP and one for "passive" to "active". An odd socket number defines a send socket and an even socket number defines a receive socket. The default HOST is the user Host to which TELNET connections are made. The default data sockets are (U+4) and (U+5) where U is the socket number used in the TELNET ICP and the TELNET connections are on sockets (U+2) and (U+3). The server has fixed data sockets (S+2) and (S+3) as well, and under normal circimstances this command and its reply are not needed.

PASSIVE (PASV)

This command requests the server-DTP to "listen" on both of his data sockets and to wait for an RFC to arrive for one socket rather than initiate one upon receipt of a transfer command. It is assumed the server has already received a SOCK command to indicate the foreign socket from which the RFC will arrive to ensure the security of the transfer.

REPRESENTATION TYPE (TYPE)

The argument specifies the representation type as described in the Section on Data Representation and Storage. Several types take a second parameter. The first parameter is denoted by a single TELNET character, as is the second Format parameter for ASCII and EBCDIC; the second parameter for local byte is a decimal integer to indicate Bytesize. The parameters are separated by a <SP> (Space, ASCII code 32.). The following codes are assigned for type:



I - Image

L # - Local byte Bytesize

The default representation type is ASCII Non-print. If the Format parameter is changed, and later just the first argument is changed, Format then returns to the Non-print default.

FILE STRUCTURE (STRU)

The argument is a single TELNET character code specifying file structure described in the Section on Data Representation and Storage. The following codes are assigned for structure:

F - File (no record structure)
R - Record structure

The default structure is File (i.e., no records).

TRANSFER MODE (MODE)

The argument is a single TELNET character code specifying the data transfer modes described in the Section on Transmission Modes. The following codes are assigned for transfer modes:

S - Stream
B - Block
C - Compressed

The default transfer mode is Stream.

FTP SERVICE COMMANDS

The FTP service commands define the file transfer or the file system function requested by the user. The argument of an FTP service command will normally be a pathname. The syntax of pathnames must conform to server site conventions (with standard defaults applicable), and the language conventions of the TELNET connection. The suggested default handling is to use the last specified device, directory or file name, or the standard default defined for local users. The commands may be in any order except that a "rename from" command must be followed by a "rename to" command and the restart command must be followed by the interrupted service command. The data, when transferred in response to FTP service commands, shall always be sent over the data connection, except for certain informative replies. The following commands specify FTP service requests:

RETRIEVE (RETR)

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

STORE (STOR)

This command causes the server-DTP to accept the data transferred via the data connection and to store the data as a file at the server site. If the file specified in the pathname exists at the server site then its contents shall be replaced by the data being transferred. A new file is created at the server site if the file specified in the pathname does not already exist.

APPEND (with create) (APPE)

This command causes the server-DTP to accept the data transferred via the data connection and to store the data in a file at the server site. If the file specified in the pathname exists at the server site, then the data shall be appended to that file; otherwise the file specified in the pathname shall be created at the server site.

ALLOCATE (ALLO)

This command may be required by some servers to reserve sufficient storage to accommodate the new file to be transferred. The argument shall be a decimal integer representing the number of bytes (using the logical byte size) of storage to be reserved for the file. For files sent with record structure a maximum record size (in logical bytes) might also be necessary; this is indicated by a decimal integer in a second argument field of the command. This second argument is optional, but when present should be separated from the first by the three TELNET characters <SP> R <SP>. This command shall be followed by a STORe or APPEnd command. The ALLO command should be treated as a NOOP (no operation) by those servers which do not require that the maximum size of the file be declared beforehand, and those servers interested in only the maximum record size should accept a dummy value in the first argument and ignore it.

RESTART (REST)

The argument field represents the server marker at which file transfer is to be restarted. This command does not cause file transfer but "spaces" over the file to the specified data checkpoint. This command shall be immediately followed by the appropriate FTP service command which shall cause file transfer to resume.

RENAME FROM (RNFR)

This command specifies the file which is to be renamed. This command must be immediately followed by a "rename to" command specifying the new file pathname.

RENAME TO (RNTO)

This command specifies the new pathname of the file specified in the immediately preceding "rename from" command. Together the two commands cause a file to be renamed.

ABORT (ABOR)

This command indicates to the server to abort the previous FTP service command and any associated transfer of data. The abort command may require "special action", as discussed in the Section on FTP Commands, to force recognition by the server. No action is to be taken if the previous command has been completed (including data transfer). The TELNET connections are not to be closed by the server, but the data connection must be closed. An appropriate reply should be sent by the server in all cases.

DELETE (DELE)

This command causes the file specified in the pathname to be deleted at the server site. If an extra level of protection is desired (such as the query, "DO you really wish to delete?"), it should be provided by the user-FTP process.

LIST (LIST)

This command causes a list to be sent from the server to the passive DTP. If the pathname specifies a directory, the server should transfer a list of files in the specified directory. If the pathname specifies a file then the server should send current information on the file. A null argument implies the user's current working or default directory. The data transfer is over the data connection in type ASCII or type EBCDIC. (The user must ensure that the TYPE is appropriately ASCII or EBCDIC).

NAME-LIST (NLST)

This command causes a directory listing to be sent from server to user site. The pathname should specify a directory or other system-specific file group descriptor; a null argument implies the current directory. The server will return a stream of

names of files and no other information. The data will be transferred in ASCII or EBCDIC type over the data connection as valid pathname strings separated by <CRLF> or <NL>. (Again the user must ensure that the TYPE is correct.)

SITE PARAMETERS (SITE)

This command is used by the server to provide services specific to his system that are essential to file transfer but not sufficiently universal to be included as commands in the protocol. The nature of these services and the specification of their syntax can be stated in a reply to the HELP SITE command.

STATUS (STAT)

This command shall cause a status response to be sent over the TELNET connection in the form of a reply. The command may be sent during a file transfer (along with the TELNET IP and Synch signals--see the Section on FTP Commands) in which case the server will respond with the status of the operation in progress, or it may be sent between file transfers. In the latter case the command may have an argument field. If the argument is a pathname, the command is analogous to the "list" command except that data shall be trasferred over the TELNET connection. If a partial pathname is given, the server may respond with a list of file names or attributes associated with that specification. If no argument is given, the server should return general status information about the server FTP process. This should include current values of all transfer parameters and the status of connections.

HELP (HELP)

This command shall cause the server to send helpful information regarding its implementation status over the TELNET connection to the user. The command may take an argument (e.g., any command name) and return more specific information as a response. The reply is type Oxx, general system status. It is suggested that HELP be allowed before entering a USER command. The server may use this reply to specify site-dependent parameters, e.g., in response to HELP SITE.

NOOP (NOOP)

This command does not affect any parameters or previously entered commands. It specifies no action other than that the server send a 200 reply.

MISCELLANEOUS COMMANDS

There are several functions that utilize the services of file transfer but go beyond it in scope. These are the Mail and Remote Job Entry functions. It is suggested that these become auxiliary protocols that can assume recognition of file transfer commands on the part of the server, i.e., they may depend on the core of FTP commands. The command sets specific to Mail and RJE will be given in separate documents.

Commands that are closely related to file transfer but not proven essential to the protocol may be implemented by servers on an experimental basis. The command name should begin with an X and may be listed in the HELP command. The official command set is expandable from these experiments; all experimental commands or proposals for expanding the official command set should be announced via RFC. An example of a current experimental command is:

Change Working Directory (XCWD)

This command allows the user to work with a different directory or dataset for file storage or retrieval without altering his login or accounting information. Transfer parameters are similarly unchanged. The argument is a pathname specifying a directory or other system dependent file group designator.

FTP REPLIES

The server sends FTP replies over the TELNET connection in response to user FTP commands. The FTP replies constitute the acknowledgment or completion code (including errors). The FTP-server replies are formatted for human or program interpretation. Single line replies consist of a leading three-digit numeric code followed by a space, followed by a one-line text explanation of the code. For replies that contain several lines of text, the first line will have a leading three-digit numeric code followed immediately by the character "-" (Hyphen, ASCII code 45), and possibly some text. All succeeding continuation lines except the last are constrained NOT to begin with three digits; the last line must repeat the numeric code of the first line and be followed immediately by a space. For example:

```
100-First Line
Continuation Line
Another Line
100 Last Line
```

It is possible to nest (but not overlap) a reply withiin a multi-line

reply. The same format for matched number-coded first and last lines holds.

The numeric codes are assigned by groups and for ease of interpretation by programs in a manner consistent with other protocols such as the RJE protocol. The three digits of the code are to be interpreted as follows:

1. The first digit specifies type of response as indicated below:

   0xx These replies are purely informative and constitute neither a positive nor a negative acknowledgment.

   1xx Informative replies to status inquiries. These constitute a positive acknowledgment to the status command.

   2xx Positive acknowledgment of previous command or other successful action.

   3xx Incomplete information. Activity cannot proceed without further specification and input.

   4xx Unsuccessful reply. The request is correctly specified but the server is unsuccessful in correctly fulfilling it.

   5xx Incorrect or illegal command. The command or its parameters were invalid or incomplete from a syntactic viewpoint, or the command is inconsistent with a previous command. The command in question has been completely ignored.

   6xx-9xx Reserved for future expansion.

2. The second digit specifies the general category to which the response refers:

   x00-x29 General purpose replies, not assignable to other categories.

   x3x Primary access. Informative replies to the "log-on" attempt.

   x4x Secondary access. The primary server is commenting on its ability to access a secondary service.

   x5x FTP results.

   x6x RJE results.

x7x Mail Portocol results.

x8x-x9x Reserved for future expansion.

3. The final digit specifies a particular message type. Since the code is designed for an automaton process to interpret, it is not necessary for every variation of a reply to have a unique number. Only the basic meaning of replies need have unique numbers. The text of a reply can explain the specific reason for that reply to a human user.

Each TELNET line delimited by a numeric code and the TELNET EOL (or group of text lines bounded by coded lines) that is sent by the server is intended to be a complete reply message. It should be noted that the text of replies is intended for a human user. Only the reply codes and in some instances the first line of text are intended for programs.

The assigned reply codes relating to FTP are:

000 Announcing FTP.
010 Message from system operator.
020 Exected delay.
030 Server availability information.
050 FTP commentary or user information.
100 System status reply.
110 System busy doing...
150 File status reply.
151 Directory listing reply.
200 Last command received correctly.
201 An ABORT has terminated activity, as requested.
202 Abort request ignored, no activity in progress.
230 User is "logged in". May proceed.
231 User is "logged out". Service terminated.
232 Logout command noted, will complete when transfer done.
233 User is "logged out". Parameters reinitialized.
250 FTP file transfer started correctly.
251 FTP Restart-marker reply.
   Text is: MARK yyyy = mmmm
      where 'yyyy' is user's data stream marker (yours)
      and mmmm is server's equivalent marker (mine)
   (Note the spaces between the markers and '=').
252 FTP transfer completed correctly.
253 Rename completed.
254 Delete completed.
257 Closing the data connection, transfer completed.
300 Connection greeting message, awaiting input.
301 Current command incomplete (no <CRLF> for long time).
330 Enter password.

331 Enter account (if account required as part of login sequence).
332 Login first, please.
400 This service not implemented.
401 This service not accepting users now, goodbye.
402 Command not implemented for requested value or action.
430 Log-on time or tries exceeded, goodbye.
431 Log-on unsuccessful. User and/or password invalid.
432 User not valid for this service.
433 Cannot transfer files without valid account. Enter account and resend command.
434 Log-out forced by operator action. Phone site.
435 Log-out forced by system problem.
436 Service shutting down, goodbye.
450 FTP: File not found.
451 FTP: File access denied to you.
452 FTP: File transfer incomplete, data connection closed.
453 FTP: File transfer incomplete, insufficient storage space.
454 FTP: Cannot connect to your data socket.
455 FTP: File system error not covered by other reply codes.
456 FTP: Name duplication; rename failed.
457 FTP: Transfer parameters in error.
500 Last command line completely unrecognized.
501 Syntax of last command is incorrect.
502 Last command incomplete, parameters missing.
503 Last command invalid (ignored), illegal parameter combination.
504 Last command invalid, action not possible at this time.
505 Last command conflicts illegally with previous command(s).
506 Last command not implemented by the server.
507 Catchall error reply.
550 Bad pathname specification (e.g., syntax error).

DECLARATIVE SPECIFICATIONS

MINIMUM IMPLEMENTATION

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

```
TYPE - ASCII Non-print
MODE - Stream
STRUCTURE - File
            Record
BYTE - 8
COMMANDS - USER, BYE, SOCK,
           TYPE, BYTE, MODE, STRU,
             for the default values
           RETR, STOR,
           NOOP.
```

The initial default values for transfer parameters are:

```
TYPE - ASCII Non-print
BYTE - 8
MODE - Stream
STRU - File
```

All Hosts must accept the above as the standard defaults.

CONNECTIONS

The server protocol interpreter shall "listen" on Socket 3. The user or user protocol interpreter shall initiate the full-duplex TELNET connections performing the ARPANET standard initial connection protocol (ICP) to server Socket 3. Server- and user- processes should follow the conventions of the TELNET protocol as specified in NIC #7104. Servers are under no obligation to provide for editing of command lines and may specify that it be done in the user Host. The TELNET connections shall be closed by the server at the user's request after all transfers and replies are completed.

The user-DTP must "listen" on the specified data sockets (send and/or receive); these may be the default user sockets (U+4) and (U+5) or a socket specified in the SOCK command. The server shall initiate the data connection from his own fixed sockets (S+2) and (S+3) using the specified user data socket and byte size (default - 8 bits). The

direction of the transfer and the sockets used will be determined by the FTP service command.

When data is to be transferred between two servers, A and B (refer to Figure 2), the user-PI, C, sets up TELNET connections with both server-PI's. He then sends A's fixed sockets, S(A), to B in a SOCK command and B's to A; replies are returned. One of the servers, say A, is then sent a PASV command telling him to "listen" on his data sockets rather than initiate an RFC when he receives a transfer service command. When the user-PI receives an acknowledgment to the PASV command, he may send (in either order) the corresponding service commands to A and B. Server B initiates the RFC and the transfer proceeds. The command-reply sequence is listed below where the messages are vertically synchronous but horizontally asynchronous:

User-PI - Server A	User-PI - Server B
C->A : ICP	C->B : ICP
C->A : SOCK HOST-B, SKT-S(B)	C->B : SOCK HOST-A, SKT-S(A)
A->C : 200 Okay	B->C : 200 Okay
C->A : PASV	
A->C : 200 Okay	
C->A : STOR	C->B : RETR

The data connection shall be closed by the server under the conditions described in the Section on Establishing Data Connections. If the server wishes to close the connection after a transfer where it is not required, he should do so immediately after the file transfer is completed. He should not wait until after a new transfer command is received because the user-process will have already tested the data connection to see if it needs to do a "listen"; (recall that the user must "listen" on a closed data socket BEFORE sending the transfer request). To prevent a race condition here, the server sends a secondary reply (257) after closing the data connection (or if the connection is left open, a "file transfer completed" reply (252) and the user-PI should wait for one of these replies before issuing a new transfer command.

COMMANDS

The commands are TELNET character string transmitted over the TELNET connections as described in the Section on FTP Commands. The command functions and semantics are described in the Section on Access Control Commands, Transfer Parameter Commands, FTP Service Commands, and Miscellaneous Commands. The command syntax is specified here.

The commands begin with a command code followed by an argument field.

The command codes are four or fewer alphabetic characters. Upper and lower case alphabetic characters are to be treated identically. Thus any of the following may represent the retrieve command:

RETR Retr retr ReTr rETr

This also applies to any symbols representing parameter values, such as A or a for ASCII TYPE. The command codes and the argument fields are separated by one or more spaces.

The argument field consists of a variable length character string ending with the character sequence <CRLF> (Carriage Return, Linefeed) for NVT-ASCII representation; for other negotiated languages a different end of line character might be used. It should be noted that the server is to take NO action until the end of line code is received.

The syntax is specified below in NVT-ASCII. All characters in the argument field are ASCII characters including any ASCII represented decimal integers. Square brackets denote an optional argument field. If the option is not taken, the appropriate default is implied.

The following are all the currently defined FTP commmands:

```
USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <acctno> <CRLF>
REIN <CRLF>
BYE <CRLF>
BYTE <SP> <byte size> <CRLF>
SOCK <SP> <Host-socket> <CRLF>
PASV <CRLF>
TYPE <SP> <type code> <CRLF>
STRU <SP> <structure code> <CRLF>
MODE <SP> <mode code> <CRLF>
RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal integer> [<SP> R <SP> <decimal integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
DELE <SP> <pathname> <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST  [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
```

NOOP <CRLF>

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

<username> ::= <string>
<password> ::= <string>
<acctno> ::= <string>
<string> ::= <char>|<char><string>
<char> ::= any of the 128 ASCII characters except <CR> and <LF>
<marker> ::= <pr string>
<pr string> ::= <pr char>|<pr char><pr string>
<pr char> ::= any ASCII code 33. through 126., printable characters
<byte size> ::= any decimal integer 1 through 255
<Host-socket> ::= <socket>|<Host number>, <socket>
<Host-number> ::= a decimal integer specifying an ARPANET Host.
<socket> ::= decimal integer between 0 and (2**32)-1
<form code> ::= N|T|C
<type code> ::= A[<SP> <form code>]|E [SP> <form code>]|I|
L <SP> <byte size>
<structure code> ::= F|R
<mode code> ::= S|B|C
<pathname> ::= <string>

SEQUENCING OF COMMANDS AND REPLIES

The communication between the user and server is intended to be an alternating dialogue. As such, the user issues an FTP command and the server responds with a prompt primary reply. The user should wait for this initial primary success or failure response before sending further commands.

Certain commands require a second reply for which the user should also wait. These replies may, for example, report on the progress or completion of file transfer or the closing of the data connection. They are secondary replies to file transfer commands.

The third class of replies are informational and spontaneous replies which may arrive at any time. The user-PI should be prepared to receive them. These replies are listed below as sponteneous.

One important group of spontaneous replies is the connection greetings. Under normal circumstances, a server will send a 300 reply, "awaiting input", when the ICP is completed. The user should wait for this greeting message before sending any commands. If the server is unable to accept input right away, he should send a 000 "announcing FTP" or a 020 "expected delay" reply immediately and a

300 reply when ready. The user will then know not to hang up if there is a delay.

The table below lists alternative success and failure replies for each command. These must be strictly adhered to; a server may substitute text in the replies, but the meaning and action implied by the code numbers and by the specific command reply sequence cannot be altered.

COMMAND-REPLY CORRESPONDENCE TABLE

COMMAND	SUCCESS	FAILURE
USER	230,330	430-432,500-505,507
PASS	230,330	430-432,500-507
ACCT	230	430-432,500-507
REIN	232,233	401,436,500-507
Secondary Reply	300	
BYE	231,232	500-505,507
BYTE	200,331	402,500-505,507
SOCK	200,331	500-505,507
PASV	200,331	500-507
TYPE	200,331	402,500-505,507
STRU	200,331	500-505,507
MODE	200,331	402,500-505,507
RETR	250	402,433,450,451,454,455,457, 500-505,507,550
Secondary Reply	252,257	452
STOR	250	402,433,451,454,455,457, 500-505,507,550
Secondary Reply	252,257	452,453
APPE	250	402,433,451,454,455,457,500-507, 550
Secondary Reply	252,257	452,453
ALLO	200,331	402,500-507
REST	200,331	500-507
RNFR	200	402,433,450,451,455,500-507,550
RNTO	253	402,433,450,451,455,456,500-507, 550
ABOR	201,202,331	500-507
DELE	254	402,433,450,451,455,500-507,550
LIST	250	402,433,450,451,454,455,457, 500-507,550
Secondary Reply	252,257	452
NLST	250	402,433,450,451,454,455,457, 500-507,550
Secondary Reply	252,257	452
SITE	200,331	402,500-507

STAT	100,110, 150,151,331	450,451,455,500-507,550
HELP	030,050	500-507
NOOP	200	500-505,507
Spontaneous Replies	000,010,020, 300,301,251,255	400,401,434-436

TYPICAL FTP SCENARIOS

TIP User wanting to transfer file from Host X to local printer:

1. TIP user opens TELNET connections by ICP to Host X socket 3.

2. The following commands and replies are exchanged:

```
TIP                          HOST X

<---------- 300 Awaiting input <CRLF>
USER username <CRLF> ---------->
<---------- 330 Enter Password <CRLF>
PASS password <CRLF> ---------->
<---------- 230 User logged in <CRLF>
SOCK 65538 <CRLF> ---------->
<---------- 200 Commmand received OK<CRLF>
RETR this.file <CRLF> ---------->
   (Host X initiates data connection to TIP socket 65538,
    i.e., PORT 1 receive)
<---------- 250 File transfer started <CRLF>
<---------- 252 File transfer completed <CRLF>
BYE<CRLF>          ---------->
<---------- 231 User logged out <CRLF>
```

3. Host X closes the TELNET and data connections.

   Note: The TIP user should be in line mode.

User at Host U wanting to transfer files to/from Host S:

In general the user will communicate to the server via a mediating user-FTP process. The following may be a typical scenario. The user-FTP prompts are shown in parentheses, '---->' represents commands from Host U to Host S, and '<----' represents replies from Host S to Host U.

LOCAL COMMANDS BY USER	ACTION INVOLVED
ftp (host) multics<CR>	ICP to Host S, socket 3, establishing TELNET connections
	<---- 330 Awaiting input <CRLF>
username Doe <CR>	USER Doe<CRLF>---->
	<---- 330 password<CRLF>
password mumble <CR>	PASS mumble<CRLF>---->
	<---- 230 Doe logged in.<CRLF>
retrieve (local type) ASCII<CR>	
(local pathname) test 1 <CR>	User-FTP opens local file in ASCII.
(for.pathname) testp11<CR>	RETR test.p11<CRLF> ---->
	Server makes data connection to
(U+4)	
	<---- 250 File transfer starts
<CRLF>	
	<---- 252 File transfer
complete<CRLF>	
type Image<CR>	TYPE I<CRLF> ---->
	<---- 200 Command OK<CRLF>
byte 36<CR>	BYTE 36<CR>LF ---->
	<---- 200 Command OK<CRLF>
store (local type) image<CR>	
(local pathname) file dump<CR>	User-FTP opens local file in Image.
(for.pathname) >udd>cn>fd<CR>	STOR >udd>cn>fd<CRLF> ---->
	<---- 451 Access denied<CRLF>
terminate	BYE <CRLF> ---->
	Server closes all connections.

\000

\032\002