Network Working Group
Request for Comments: 354
NIC: 10596
Categories D.4, D.5, D.7
Obsoletes: RFC 264 and 265

Abhay Bhushan
MIT-MAC

July 8, 1972

THE FILE TRANSFER PROTOCOL

I. INTRODUCTION

The File Transfer Protocol (FTP) is a protocol for file transfer betweet HOSTs (including terminal IMPs), 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) fo 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 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, elegant, and easily implemented protocol design.

This paper assumes knowledge of the following protocols:

1) The HOST-HOST Protocol (NIC #8246)

2) The initial Connection Protocol (NIC #7101)

3) The TELNET Protocol (NWG/RFC #318, NIC #9348)

II. 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.

11.A. Terminology

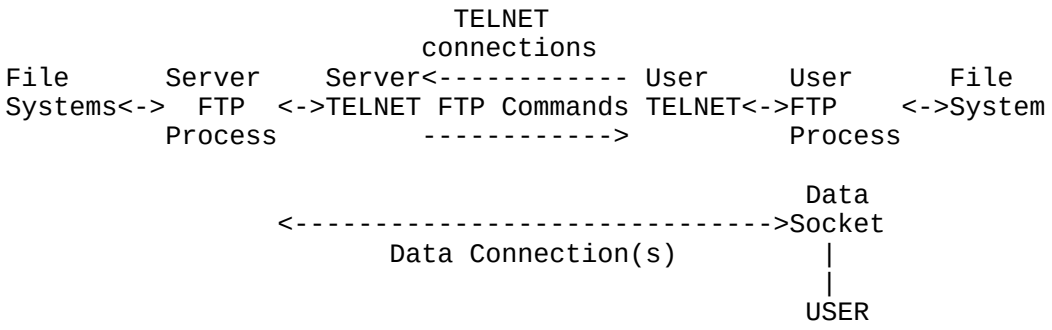
ASCII	The USASCII character set as defined in NIC #7104. In FTP, ASCII characters are defined to be the lower half of an eight bit code set (i.e., the most significant bit es zero).
access controls	Access controls define users' access privileges to the use of a system, and to the files in that system. Access controls are necessary to prevent unauthorized or accidental use of files. It is the prerogative of a user-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. Data connection byte size is not necessarily the byte size in which data is to be stored in a system, and may not be related to the structure of 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 data connection may be in either direction (server-to-user or user-to server).
data socket	The socket on which a User-FTP process "listens" for a data connection.
EOF	The end-of-file conidition 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 form 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 including EOR and EOF. The transfer modes defined in FTP are described in Section III.A.
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 NFS 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 hte 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 are not mandatory.
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 an ASCII text string. The codes are for use by programs and the text is for human users.
server-FTP process	A process or set of processes which perform the function of file transfer in cooperation with a user-FTP process. The server-FTP process must interpret and respond to user commands and initiate the data connection.

server site	A HOST site wich has a server-FTP process.
server-TELNET	A TELNET process which listens on a specified socket for an ICP initiated by a user-TELNET, and perform in accordance with the ARPANET TELNET Protocol.
TELNET connections	The full-duplex communication path between a user-TELNET and a server-TELNET. 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 Section III.B.
user	A process on behalf of a human being or a human being wishing to obtain file transfer service.
user site	A HOST site satisfying any of the following conditions: 1) The site where a user is located, 2) a site where a user-FTP process is located, 3) a site to which a data connection is made by a server. In the normal case, the sites defined by 1, 2, and 3 are the same site, but nothing in FTP requires that this be so.
user-FTP process	A process or set of precesses which perform the function of file transfer in cooperation with a server-FTP process. The user-FTP process 1) initiates the ICP (via a user-TELNET, 2) initiates FTP commands and 3) "listens" on the data socket for the data connection. In some obvious cases (use from TIPs and other mini-HOSTs) a user-FTP process will be subsumed under the term "user".
user-TELNET	A TELNET process which initiates an ICP to a specified server-TELNET socket, and performs in accordance with the ARPANET TELNET protocol.

II.B. 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.

3. The distinctions between user-FTP and user-TELNET, and between server-FTP and server-TELNET may not be as clear-cut as shown above. For example, a user-TELNET may be directly driven by the user.

FIGURE 1 Model for FTP Use

In the model described in Figure 1, the user-TELNET initiates the TELNET connection. Standard FTP commands are then generated by the user and transmitted to the server site via the TELNET connections. FTP commands are in ASCII, in accordance with NVT conventions and the TELNET protocol. Note that commands may be initiated by the user directly through the user-TELNET or via a user-FTP process. Standard replies are sent from the server to the user in response to the commands over the TELNET connections.

The FTP commands specify the parameters for teh data connection (data socket, byte size, transfer mode, and representation type), and the nature of file system operation (store, retrieve, append, delete, etc.). The user-FTP process or its designate should "listen" on

the specified data socket, and it is the server's responsibility to initiate the data connection and data transfer in accordance with the specified data connection parameters. It should be noted that the data socket need not be in the same HOST that initiates the FTP commands via the TELNET connection, but the user or his user-FTP process must ensure a "listen" on the specified data socket. A practical example of such file transfer to third HOSTs is a maxi-HOST user (who may actually be a TIP user) whishing to transmit a file to or from an I/O device attached to a TIP. It should also be noted that two data connections, one for send and the other for receive, may exist simultaneously.

The protocol requires that the TELNET connections be open while data transfer is in progress. It is the responsibility of the user to close the TELNET connections when finished using the FTP service. The server may abort data transfer if the TELNET connections are closed.

III. DATA TRANSFER FUNCTIONS

Data and files are transferred only via the data connection. The data transfer of data is governed by FTP data transfer commands received on the TELNET connections. The data transfer functions include establishing the data connection to the specified data socket in the specified HOST (using the specified byte size), transmitting and receiving data in the specified representation type and transfer mode, handling EOR and EOF conditions and error recovery (where applicable).

III.A Establishing Data Connection

The user site shall "listen" on the specified data socket. The FTP request command determines the direction of data transfer, and the socket number (odd or even) which is to be used in establishing the data connection. The server on receiving the appropriate store or retrieve request shall initiate the data connection to the specified user data socket in the specified byte size (default byte size is 8 bits) and send a reply indicating that file transfer may proceed. Prior to this the server should send a reply indicating the server socket for the data connection. The user may use this server socket information to ensure the security of his data transfer. The server may send this reply either before of after initiating the data connection.

The byte size for the data connection is specified by the TYPE (ASCII is 8 bits), or TYPE and BYTE commands. It is not required by the protocol that servers accept all possible byte size. The user of various byte size is for efficiency in data transfer and servers may implement only those byte size for which their data transfer is efficient. It is however recommended that servers implement at least the byte size of 8 bits.

After the data transfer is completed, it is the server's responsibility to close the data connection except when the user is sender of data. The data connection shall be closed under any of the following conditions:

1) server receives an abort command form user.

2) EOF in stream mode indicated by closing data connection.

3) the socket or byte size specification is changed.

4) any of the TELNET connections are closed.

5) an irrecoverable error condition.

It should be noted that two simultaneous data connections (for send and receive) may exist. It is a server option, however, to close the data connection after each instance of file transfer.

III.B Data Representation and Storage

Data is transferred from a storage device in sending HOST to a storage device in receiving HOST. Often it is necessary to perform certain transformations on the data because data storage representations in the two systems are different. For example, NVT-ASCII has different data storage representations in different systems. PDP-10's generally store NVT-ASCII as five 7-bit ASCII characters, left justified in a 36 bit word. 360's store NVT-ASCII as 8-bit EBCDIC codes. Multics stores NVT-ASCII as four 9-bit characters in a 36-bit word. It may desirable to convert characters into the standard NVT-ASCII representation when transmitting text between disimilar systems. The sending and receiving site 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 length. 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) o 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 types reprentations. Transformations desired beyond this limited capability should be performed by the user directly or via the use of the Data Reconfiguration Service (DRS, RFC #138, NIC #6715). Additional representation types may be defined later if there is a demonstrable need.

Data representations are handled in FTP by a user specifying a representation type. The type may also specify a fixed transfer byte size. For example in ASCII and Print File representations, the transfer byte size must be 8 bits. Only in the Image and Local Byte representations the byte size specified by the BYTE command is to be used. The following data representation types are currently defined in FTP:

1. ASCII — The sender converts data form its internal character representation to the standard ARPANET ASCII form. The receiver converts the data from the standard form to its own internal form. The data is transferred in the standard form. The transfer byte size must be 8 bits. This type would be used for transfer of text files. This is be default type, and it is recommended that this type be implemented by all.

2. Image — The sender transforms data from contiguous bits to bytes for transfer. The receiver transforms the bytes into bits, storing them contiguously independent of the byte size chosen for data transfer. Typical uses for the Image type are transfer of executable programs between like machines, and transfer of binary (non-text) data. It is recommended that this type be implemented by all for some byte size preferably including the 8 bit byte size.

3. Local Byte — This representation allows for efficient storage, use, and retrieval of data. The mann in which data is to be transformed depends on the byte size for data transfer, and the particular HOST being used. The transformation scheme for different byte size is to be well publicized b all server sites. This transformation shall be invertible (i.e., if a file is stored using a certain transfer byte size, an identical file must be retrievable using the same byte size and representation type). It is the user’s responsibility to keep track of the representation type and byte size used for his transfer. Typical uses of the Local Byte type are in efficient storage and retrieval of files, and transfer of structured binary

	data. This type may be identical to the image type for byte size which are integral multiples of or factors of the computer word length-
4. Print File-ASCII	The server site will transform the ASCII file in a form suitable for printing at the server site. The byte size must be 8 bits. The transformation may not be invertible. This type is different from ASCII in that TABs, FFs and other ASCII format effector characters may be replaced by SPs, LFs, and other substitute characters. The print file conversions are to be well publicized by all server sites. This type would be used when the file is destined for an ASCII printer. This type in some systems may be identical to the ASCII type. It is recommended that this type be implemented by all.
5. EBCDIC Print File	The server site will transform the EBCDIC file into a form suitable for printing at the server site. The byte size must be 8 bits. the transformation may not be invertible. This type would be used when the file is destined for an EBCDIC printer. Only systems which use EBCDIC for their internal character representation need accept this type.

It should be noted that a serving HOST need not accept all representation types and/or byte size, but it must inform the user of the fact by sending an appropriate reply.

III.C File Structure and Transfer Modes

The only file structures supported directly in FTP at the present time are record structures. However, the use of record structures is not mandatory. A user with no record structure in his file should be able to store and retrieve his file at any HOST. A user wishing to transmit a record structured file must send the appropriate FTP 'STRU' command (the default assumption is no record structure). A serving HOST need not accept record structures, but it must inform the user of this fact by sending an appropriate reply. Any record structure information in the data stream may subsequently be discarded by the receiver.

All data transfer must end with an EOF. The EOF is defined by the data transfer mode. For files that have record structures, an EOR is also defined by the transfer mode. Only the transfer modes and representation type combinations that have EOR defined may be used for transfer or files with record structures. Records may be of zero length but they must be contained in file boundaries. The relationship between files and records is heirarchical and an EOF implies an EOR.

The following data transfer modes are defined in FTP:

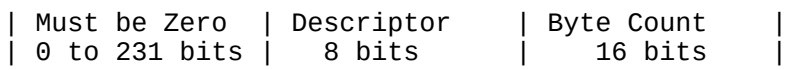
1. Stream — The file is transmitted as a stream of bytes of the specified byte size. The EOF is signalled by closing the data connection. Any representation type and byte size may be used in the stream mode but record structures are possible only with the ASCII representation type. The convention is that the ASCII character CR (Carriage Return, Code 13.) followed by LF (Line Feed, Code 10.) Indicates an EOR in stream mode and ASCII representation type. This is the default mode, and it is recommended that this mode be implemented by all.

2. Text — The file is ASCII text transmitted as sequence of 8-bit bytes in the ASCII representation type. Record structures are allowed in this mode. The EOR and EOF are defined by the presence of special "TELNET-control" codes (most significant bit set of one) in the data stream. The EOR code is 192 (octal 300, hex C0). The EOF code os 193 (octal 301, hex C1). The byte size for transfer is 8 bits.

3. 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 file block (EOF), last record block (EOR), restart marker (see section III.D), or suspect data (i.e. the data being transferred is suspected of errors and is not reliable). Record structures are allowed in this mode, and any representation type or byte size may be used. The header consists of integral number of bytes whose length is greater than or equal to 24 bits. Only the least significant 24 bits (right-jusified) of header shall have

information, other must significant bits must be zero. 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 data bytes > 24



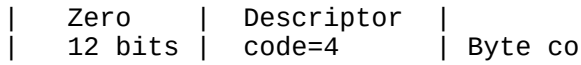
The following descriptor codes are assigned:

Code	Meaning
0	An ordinary block of data.
1	End of data block is EOR.
2	End of data block is EOF.
3	Suspected errors in data block.
4	Data block is a restart marker.

The restart marker is imbedded in the data stream as integral number of 8-bit bytes (representing printable ASCII characters) right-justified in integral number of data bytes greater than 8 bits. For example if the byte size is 7 bits, the restart marker byte would be one byte right-justified per two 7-bit bytes as shown below:

Two 7-bit bytes

For byte size of 16 bits or more, two more marker bytes shall be packed right-justified. The end of the marker may be delimited by the character SP (code 32.). If marker characters do no exactly fit an integral byte, the unused character slots should contain the ASCII character SP (code 32.). For example, to transmit a six character marker in a 36-bit byte size, the following three 36.bit bytes would be sent:



	Marker 8 bits	Marker 8 bits	Marker 8 bits	Marker 8 bits
	Marker 8 bits	Marker 8 bits	SP 8 bits	SP 8 bits

4 Hasp — The file is transmitted as a sequence of 8-bit bytes in the standard Hasp-compressed data format (document to be issued by Bob Braden, UCLA). This mode achieves considerable compression of data for print files. Record structures are allowed in the Hasp mode.

III.D 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 user from system failures (such as failure of either HOST, FTP-process, or the IMP subnet).

The restart procedure is defined only for the block mode of data transfer. It requires the sender of data to insert a special marker code in teh data stream with some marker information. The marker information has meaning only to the sender, but must consist of printable ASCII characters. The printable ASCII characters are defined to be codes 33. through 126. (i.e., not including codes 0. through 31. and the characters SP and DEL). The marker could represent a bit-count,a record-count, or any other information by wich 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 examples Illustrate the use of the restart procedure.

1. When server is the sender of data, the server-FTP process inserts an appropriate marker block in the data stream at a convenient data point. The user-FTP process receiving the data, marks the coressponding data point in its file system and conveys the last known sender and receiver marker information to the user. In the event of system failure, the user or user-FTP process restarts the server at the

last server marker by sending a restart command with the server's marker code at its argument. The restart command is transmitted over the TELNET connection and is immediately followed by the command (such as store or retrieve) which was being executed when the system failure occured.

2. When user is the sender of data, the user-FTP process inserts the appropriate marker block in the data stream. The server-FTP process receiving the data, marks the corresponding data point in its file system. The server does not store this marker but conveys the last known sender and receiver marker information to the user over the TELNET connections by appropriate reply codes. The user or the user-FTP process then restarts transfer in a manner identical to that described in the first example.

IV. FILE TRANSFER FUNCTIONS

The TELNET connections on which FTP commands and replies are transmitted, are initiated by the user-FTP process via an ICP to a standard server socket. FTP commands are then transmitted from user to server, and replies are transmitted from server to user. The user file transfer functions invoive sending the FTP commands, interpreting the replies received and transferring data over the data connection in the specified manner. The server file transfer functions involve accepting and interpreting FTP commands, sending replies, setting up the data connection, and transferring data.

IV.A FTP Commands

FTP commands are ASCII terminated by the ASCII character sequence CRLF (Carriage Return follow by Line Feed). The command codes themselves are ASCII alpabetic characters terminated by the ASCII character 'space' (code = 32.). For convenience, the command codes are defined to be four (or less) ASCII alphanumeric characters (including both upper and lower case alphabetic characters). The command codes and the semantics of commands are described in this section, but the detailed syntax of commands is specified in Section V.b, the reply sequence are discussed in Section V.C, and scenarios illustrating the use of commands are provided in Section V.D.

FTP commands may be partitioned as those specifying access-control identifiers, data transfer parameters, or FTP service requests.

IV.A.1 Access Control Commands

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

User name (USER) - The argument field is an ASCII string identifying the user. The user identification is that wich 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 pasword command may also be required by some servers.

Password (PASS) - The argument field is an ASCII string identifying the user's password. This command must be immediately preceded by the user name command, and together it completes the user's identifecation for access control.

IV.A.2 Data Transfer 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 teh last specified value, or if no value has been specified, the standard default value specified 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 an ASCII-represented decimal integer (1 through 255), specifying the byte size for the data connection for local byte and image representation types. The default byte size is 8 bits. The byte size is always 8 bits in the ASCII and Print file representation types. A server may reject specific byte size/type combinations by sending an appropriate reply.

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 from server to user and the other for user to server data transfer. 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).

Representation Type (TYPE) - The argument is a single ASCII character code specifying the representation types described in section III.B. The following codes are assigned for type:

A - ASCII
I - Image
L - Local Byte
P - Print file in ASCII
E - EBCDIC print file

The default representation type is ASCII

File Structure (STRU) - The argument is a single ASCII character code specifying file structure described in section III.C. 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 ASCII character code specifying the data transfer modes described in Section III.C. The following codes are assigned for transfer modes:

S - Stream (bytes, close is EOF)
B - Block (Header with descriptor and count)
T - Text (TELNET control mode for EOR, EOF)
H - Hasp (specially formatted compressed data)

The default transfer mode is Stream.

IV.A.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), except that ASCII characters must be used (in conformance with the TELNET Protool). 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 some servers may require an "allocate" command before a "store" command. The data when transferred in response to FTP service commands shall always be over the data connection. The following commands specify FTP service requests:

Retrieve (RETR) - This command achieves the transfer of a copy of file specified in pathname, from server to user site. The status and contents of a file at server site shall be unaffected.

Store (STOR) - This command achieves the transfer of a copy of file from user to server site. If file specified in pathname exists at the server site, then its contents shall be replaced by the contents of the file being transferred. A new file is created at the server site if the file specified in pathname does not already exist.

Append (with create) (APPE) - This command achieves the transfer of data from using to serving site. If file specified in pathname exists at the server site, then the data transferred shall be appended to that file, otherwise the file specified in pathname shall be created at the server site.

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.

Delete (DELE) - This command causes teh file specified in 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 server to user site. If pathname specifies a directory, the server should transfer a list of files in the specified directory. If pathname specifies a file then server should send current information on the file. This command may be used to obtain the contents of a file directory (the response should be sent in ASCII type) or test the existence of a file and its current status.

Allocate (ALLO) - This command my be required by some servers to reserve sufficient storage to accomodate the new file to be transferred. The command field shall be a decimal integer representing the number of bytes (of size specified by the byte size command) of storage to be reserved for the file. This command shall be followed by a store or append command. The ALLO command should be treated as a NO-OP (no operation) by thuse servers which do not require that the maximum size of the file be declared beforehand.

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.

Status (STAT) - This command shall cause a status response to be sent over the TELNET connection in form of a reply. The command may have an argument field such as a pathname. if the argument is a pathname, the command is analogous to the "list" command except that data shall be transferred in ASCII on the TELNET connection. If no argument is specified, the server should return general status information about the server FTP process. This may include service availability, the current settings for the relevant FTP parameters (including default settings), and the status of command execution and connections.

Abort (ABOR) - This command indicates to the server to abort the previous FTP service command and any associated transfer of data. The abort command should be preceded by the TELNET SYNCH condition (indicated by the combination of the DATA MARK and the INS). No action is to be taken if the previous command has been completed (including data transfer). The TELNET connections is not to be closed by the server, but the data connection may be closed. An appropriate reply should be sent by the server.

Logout (BYE) - This command terminates a USER and if file transfer is not in progress, closes the TELNET connection. If file transfer is in progress, the connection will remain open for result response and will then close. During the interim a new USER command (and no other command) is acceptable.

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

IV.B FTP Replies

The server sends FTP replies to user over the TELNET connections in response to FTP commands. The FTP replies constitute the acknowledgement or completion code (including errors). The FTP-server replies are formatted for human or program interpretation. The replies consist of a leading three digit numeric code followed by a space followed by a text explanation of the code. 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 interpredet as follows:

a) The first digit specifies type of response as indicated below:

   000 These replies are purely informative and constitute neither a positive nor a negative acknowledgement.

   1xx informative replies to status inrequiries. 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 corretly fulfilling it.

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

   6xx - 9xx Reserved for future expansion.

b) The second digit specifies the general category to which the response refers:

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

   x30 Primary access. Informative replies to the "log-on" attempt.

   x40 Secondary access. The primary server is commenting on its ability to access a secondary service.

   x5x FTP results.

   x6x RJE resutls.

   x7x-x9x Reserved for future expansion.

c) 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 (ended by CRLF) from the server is intended to be a complete reply message. if it is necessary to continue the text of a reply onto following lines, then those continuation replies contain the special reply code of three spaces. It should be noted that text of replies are 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 General information message (site, time of day, etc.)
030 Server availibility information.
050 FTP commentary or user information.
100 System status reply.
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.
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.
255 FTP server data socket reply
Text is: SOCK nnnn
where nnnn is decimal integer representing
the server socket for data connection.
300 Connection greeting message, awaiting input.
301 Current command incomplete (no CRLF for long time).
330 Enter password (may be sent with hide-your-input).
400 This service not implemented.
401 This service not accepting users now, goodbye.
430 Log-on time or tries exceeded, goodbye.
431 Log-on unsuccessful. User and/or password invalid.
432 User not valid for this service.

434 Log-out forced by operator action. 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.
500 Last command line completely unrecognized.
501 Syntax of last command in 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 Requested action not implemented by the server.

V. DECLARATIVE SPECIFICATIONS

V.A. Connections

The server-FTP process at the server site shall "listen" on Socket 3, via its server-TELNET. The user or user-FTP process at the user site shall initiate the full-duplex TELNET connections via its user-TELNET performing the ARPANET standard initial connection protocol (ICP) to server socket 3. The TELNET connections shall be closed by the user site upon completion of use.

The user site shall "listen" on the specified data socket or sockets (a send and/or a receive socket). The server site shall initiate the data connection using the specified data socket and byte size. The direction of data connection and the data socket used shall be determined by the FTP service command. The server shall send a reply to the user indicating the server data socket so that the user may ensure the security of data transfer. This can be done at any time prior to the first transfer of data over a data connection.

The data connection shall be closed by the server site under the conditions described is Section III.A. The server should in general send a reply before closing the data connection to avoid problems at the user end.

V.B. Commands

The commands are ASCII character strings transmitted over the TELNET connections as described in section IV.A. The command functions and semantics are described in sections IV.A.1, IV.A.2, IV.A,3, and IV.A.4. The command syntax is specified here.

The commands begin with a command code followed by an argument field. The command codes are four of less ASCII 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

The command codes and the argument fields are separated by one or more spaces.

The argument field consists of a variable length ASCII character string ending with the character sequence CRLF (Carriage Return immediately followed by Line Feed). In the following section on syntax it should be stressed that all characters in the argument field are ASCII characters. Thus a decimal integer shall mean an ASCII represented decimal integer.

The following are all the currently defined FTP commands:

```
USER <user name> CRLF
PASS <password> CRLF
BYTE <byte size> CRLF
SOCK <HOST-socket> CRLF
TYPE <type code> CRLF
STRU <structure code> CRLF
MODE <mode code> CRLF
RETR <pathname> CRLF
STOR <pathname> CRLF
APPE <pathname> CRLF
RNFR <pathname> CRLF
RNTO <pathname> CRLF
DELE <pathname> CRLF
LIST <pathname> CRLF
ALLO <decimal integer> CRLF
REST <marker> CRLF
STAT <pathname> CRLF
ABOR <empty> CRLF
 Bye <empty> CRLF
```

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

```
 <username> ::= <string>
 <password> ::= <string>
 <string> ::= <empty> | <char> | <char><string>
 <char> ::= any of the 128 ASCII characters except CR and LF.
 <marker> ::= <pr string>
```

```
<pr string> ::= <empty> | <pr char> | <pr char><pr string>
<pr char> ::= any ASCII code 33 through 126.
<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 o and (2**32)-1
<type code> ::= A|I|L|P|E
<structure code> ::= F|R
<mode code> ::= S|B|T|H
<pathname> ::= <string>
<decimal integer> ::= <digit> | <digit><decimal integer>
<digit> ::= 0|1|2|3|4|5|6|7|8|9|
<empty> ::= the null string (specifies use of default).
```

V.C 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 of failure response before sending further commands.

A second type of reply is sent asynchronously with respect to user commands. These replies may for example report on the progress or completion of file transfer and as such are secondary replies to file transfer commands.

The third class of replies are informational and spontaneous replies which may arrive at any time. These replies are listed below as spontaneous.

COMMAND-REPLY CORRESPONDENCE TABLE

COMMAND	SUCCESS	FAIL
USER	230,330	430-432,500-505
PASS	230	430-432,500-505
BYE	231,232	430-432,500-505
BYTE	200	500-506
SOCK	200	500-506
TYPE	200	500-506
MODE	200	500-506
RETR	250	450,451,500-506
Secondary Reply	252	452
STOR	250	451,451,500-506
Secondary Reply	252	452,453
APPE	250	451,451,500-506
Secondary Reply	252	452,453

RNFR	200	450,451,500-506
RNTO	253	450,451,500-505
DELE	254	450,451,500-506
LIST	250	450,453,500-506
Secondary Reply	252	452
ALLO	200	500-506
STAT	100,150,151	450,451,500-506
REST	200	500-506
ABOR	201,202	500-505
Spontaneous Replies	0xx,300,301 251,255	400,401,434-436

V.D. Tyical FTP Scenarious

1. TIP User wanting o transfer file from FOST X to local printer:

   a) TIP user opens TELNET connections by ICP to HOST X, socket 3.

   b) The following commands and replies are exchanged:

```
       TIP                              HOST X

       USER username CRLF  -------->
       <-------  330 Enter Password CRLF

       PASS password CRLF  -------->
       <-------  230 User logged in CRLF

       SOCK 65538 CRLF     -------->
       <-------  200 Command received OK CRLF

       RETR this.file CRLF -------->
       <-------  255 SOCK 5533 CRLF

       (HOST X initiates data connection to
        TIP socket 65538, i.e., PORT 1 receive)

       <------- 250 File transfer started

       BYE CRLF            ------_->
       <------- 252 File transfer completed
```

   c) HOST X closes the TELNET and data connections.

   Note: The TIP user should be in line mode and can thus use
         local TIP editing such as character delete.

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

In general the user would communicate to the server via a mediating user-FTP process. The following may be a typical scenario. The user-FTP prompts are shown in parenthesis, '---->' 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.
username Doe CR	USER DoeCRLF ----> <---- 330 passwordCRLF
password mumble CR	PASS mumbleCRLF ----> <---- 230 Doe logged in.CRLF
retrieve (local type ASCIICR (local pathname) test 1 CR (for. pathname) test.pl1CR	USER-FTP open local file in ASCII. RETR test.pl1 CRLF ----> <---- 255 SOCK 1233CRLF Server makes data connection to (U+4). <---- 250 File transfer startsCRLF <---- 252 File transfer completeCRLF
type imageCR	TYPE |CRLF ----> <---- 200 Command OKCRLF
byte 36CR	BYTE 36CRLF ----> <---- 200 Command OKCRLF
store (local type) ImageCR (local pathname) file dumpCR (for. pathname) >udd>cn>fdCR	User-FTP opens local file in Image. STOR >udd>cn>fdCRLF ----> <---- 451 Access deniedCRLF
terminate	BYECRLF <---- 231 Doe logged outCRLF Server closes all connections.

ACKNOWLEDGEMENTS

The work on file transfer protocol has involved many people. This document reports the work of a group rather than the author alone. The author gratefully acknowledges the conributions of the following:

Bob Braden	UCLA-CCCN
Arvola Chan	MIT-MAC
Bill Crowther	BBN-TIP
Eric Harslem	RAND
John Heafner	RAND
Chuck Holland	UCSD
Alex McKenzie	BBN (NET)
Bob Metcalfe	XPARC
Jon Postel	UCLA
Neal Ryan	MIT-MAC
Bob Sundberg	HARVARD
Ray Tomlinson	BBN (TENEX)
Dick Watson	SRI-ARC
Jim White	SRI-ARC
Richard Winter	CCA

[This RFC was put into machine readable form for entry]
[into the online RFC archives by Gottfried Janik 9/97]