Network Working Group
Request for Comments #172
NIC 6794
Categories: D.4, D.5, and D.7
Updates: 114
Obsolete: None

23 June 1971
Abhay Bhushan, MIT
Bob Braden, UCLA
Will Crowther, BBN
Eric Harslem, Rand
John Heafner, Rand
Alex McKenzie, BBN
John Melvin, SRI
Bob Sundberg, Harvard
Dick Watson, SRI
Jim White, UCSB

THE FILE TRANSFER PROTOCOL

I. INTRODUCTION

The file transfer protocol (FTP) is a user-level protocol for file transfer between host computers (including terminal IMP's), on the ARPA computer network. The primary function of FTP is to facilitate transfer of files between hosts, and to allow convenient use of storage and file handling capabilities of other hosts. FTP uses the data transfer protocol described in RFC 171 to achieve transfer of data. This paper assumes knowledge of RFC 171.

The objectives of FTP are to promote sharing of files (computer programs and/or data), encourage indirect use (without login or implicit) of computers, and shield the user from variations in file and storage systems of different hosts, to the extent it is practical. These objectives are achieved by specifying a standard file transfer socket and initial connection protocol for indirect use, and using standard conventions for file transfer and related operations.

II. DISCUSSION

A file is considered here to be an ordered set of arbitrary length, consisting of computer (including instructions) data. Files are uniquely identified in a system by their pathnames. A pathname is (loosely) defined to be the data string which must be input to the file system by a network user in order to identify a file. Pathname usually contains device and/or directory names, and file names in case of named files. FTP specifications provide standard file system commands, but do not provide standard naming convention at this time. Each user must follow the naming convention of the file system he wishes to use. FTP may be extended later to include standard conventions for pathname structures.[1]

A file may or may not have access controls associated with it. The access controls designate the users' access privilege. In the absence of access controls, the files cannot be protected from accidental or unauthorized usage. It is the prerogative of a resident file system to provide protection, and selective access. FTP only provides identifier and password mechanisms for exchange of access control information. It should however be noted, that for file sharing, it is necessary that a user be allowed (subject to access controls) to access files not created by him.

FTP does not restrict the nature of information in the file. For example, a file could contain ASCII text, binary data computer program, or any other information. A provision for indicating data structure (type and byte size) exists in FTP to aid in parsing, interpretation, reconfiguration, and storage of data. To facilitate indirect usage, the cooperating file transfer processes may be disowned "daemon" processes

which "listen" to agreed-upon sockets, and follow the standard initial connection protocol for establishing a full-duplex connection. It should be noted that FTP could also used directly by logging into a remote host, and arranging for file transfer over specific sockets.

FTP is readily extensible, in that additional commands and data types may be defined by those agreeing to implement them. Implementation of a subset of commands is specifically permitted, and an initial subset for implementation is recommended.[2] The protocol may also be extended to enable remote execution of programs, but no standard procedure is suggested.

For transferring data, FTP uses the data transfer protocol specified in RFC 171. As the data transfer protocol does not specify the manner in which it is to be used by FTP, implementation may vary at different host sites. Hosts not wishing to separate the data transfer and file transfer functions, should take particular care in conforming to the data transfer protocol specifications of RFC 171.

It should be noted, that FTP specifications do not require knowledge of transfer modes used by data transfer protocol. However, as file transfer protocol requires the transfer of more than a single control transaction over the same connection, it is essential that hosts be able to send control transactions in either 'transparent block' (type B9) of 'descriptor and counts' (type BA) modes. (Type BB, the indefinite bit stream mode is not suitable as it limits transfer to singles transactions.).

The use of data transfer aborts (type B6) is neither required, nor defined in FTP. FTP has its own error terminate which may be used to abort a file transfer request. FTP also does not define the structure of files, and there are no conventions on the use of group, record and unit separators.[3] A file separator is, however, used to indicate the end of file. It is strongly recommended that default options be provided in implementation to facilitate use of file transfer service. For example, the main file directory on disk, a pool directory, user directory or directory last accessed could serve as standard pathname defaults. Default mechanisms are convenient, as the user doesn't have to specify the complete pathname each time he wishes to use the file transfer service.

III. SPECIFICATIONS

1. Data Transfer

FTP uses the data transfer protocol described in RFC 171, for transferring data and/or control transaction. Both data and control transactions are communicated over the same connection.

2. Data Transactions

Data transactions represent the data contained in a file. There is no data type or byte size information contained in data transactions. The structure of data is instead communicated via control transactions. A file may be transferred as one or more data transactions. The protocol neither specifies nor imposes any limitations on the structure (record, group, etc) or length of file. Such limitations may however be imposed by a serving host. The end of a file may be indicated either by a file separator (as defined in data transfer protocol), or by closing connection (in type B0). In particular a serving or using host should not send the ETX, or other end of file character, unless such a character is part of the data in file (i.e., not provided by system).

3. Control Transactions

The control transactions may be typified as requests, identifiers, and terminates. A request fulfillment sequence begins with a request and ends with receipt of data (followed by End-of-File) or a terminate.

3A. Op Codes

Control transactions are distinguished by their first byte referred as op code. A standard set of opcodes is defined below. Implementation of a workable[4] subset of opcodes is specifically permitted. Additional standard opcodes may be assigned later. Opcodes hex 5A (octal 100) through hex FF (octal 377) are for experimental use.

Op Code Hex	Op Code Octal	Operation
00	000	Change data type identifier
01	001	Retrieve Request
02	002	Store request (replaced if file already exists)
03	003	Delete request
04	004	Rename_from request
05	005	Rename_to request
06	006	List_file_directory request
07	007	Username identifier
08	010	Password identifier
09	011	Error or unsuccessful terminate
04	012	Acknowledge or successful terminate
0B	013	Append request (add to existing file)
0C	014	
through	through	Reserved for standard assignment
4F	077	
5A	100	
through	through	Reserved for experimental use
FF	377	

3B. Syntax and Semantics

3B.1 Data Types

The ’change data type’ control transactions identifies the structure of data (data type and byte size) in succeeding data transactions. This transaction shall contain two more bytes in addition to the opcode byte. The first of these bytes shall convey a data type or code information and the second byte may convey the data byte size, where applicable. This information may be used to define the manner in which data is to be parsed, interpreted, reconfigured or stored. Change data type need be sent only when structure of data is changed from the preceding.

Although, a number of data types are defined, specific implementations may handle only limited data types or completely ignore the data type and byte size descriptors. Even if a host process does not "recognize" a data type, it must accept data (i.e., there is no such thing as a data type error.) These descriptors are provided only for convenience, and it is not essential that they be used. The standard default is to assume nothing about the information and treat it as a bit stream (binary data, byte size 1)[5] whose interpretation is left to a higher level process, or the user.

---

* It is, however, possible that this bit stream is treated like ASCII characters in specific instances such as transmitting a file to a line printer.

The following data type codes are currently assigned. Where a byte size is not implicit in data type, it may be provided by the second byte.

Hex	Octal		
00	000	1	Bit stream (standard default)
01	001	none	Binary data bytes
02	002	8	Network ASCII characters
03	003	8	EBCDIC characters
04	004	36	DEC-packed ASCII (five 7-bit characters, 36th bit 1 or 0)
05	005	8	Decimal numbers, net. ASCII
06	006	8	Octal numbers, net. ASCII
07	007	8	Hexadecimal numbers, net. ASCII
08	010		
through	through		Reserved for standard assignment
4F	077		
5A	100		
through	through		Reserved for experimental use
FF	377		

3B.2 Requests and Identifiers

Retrieve, delete, name_from, rename_t, and append requests contain a pathname, following the op code, in the information field. A pathname may also follow the opcode in list_file_directory request.

A pathname must uniquely identify a file in the serving host. The syntax of pathnames and identifying information shall conform to serving host conventions, except that standard network ASCII (as defined in the TELNET protocol) shall be used.

The store request has a 4-byte (32 bits) 'allocate size' field followed by pathname. 'Allocate size' indicates the number of bits of storage to be allocated to the file. A size of zero indicates that server should use his default.

Retrieve request achieves the transfer of a copy of file specified in pathname, from serving to using host. The status and contents of file in serving host should be unaffected.

Store request achieves the transfer of a copy of file specified in pathname, from using to serving host. If file specified in pathname exists on serving hosts, then its contents shall be replaced by the contents of the file being transferred. A new file is created at the serving host, if the file specified in pathname does not exist.

Append request achieves the transfer of data from using to serving host. The transferred data is appended to file specified in pathname, at serving host.

Rename-from and rename-to requests cause the name of the file specified in pathname of rename_from to be changed to the name specified in pathname of rename_to. A rename_from must always be followed by a rename_to request.

Delete request causes file specified in pathname to be deleted from the serving host. If an extra level of protection is desired such as the query "Do you really wish to delete this file?", it is to be a local implementation in the using system. Such queries should not be transmitted over network connections.

Username and password identifiers contain the respective identifying information. Normally, the information will be supplied by the user of the file transfer service. These identifiers are normally sent at the start of connection.

3B.3 Error and Acknowledge Terminates

The error transactions may have an error code indicated by the second descriptor byte. Transmission of an error message in text is also permitted. The following error codes are defined.

Error Code (2nd descriptor byte)		Meaning
Hex	Octal	
00	000	Error condition indicated by computer system (external to protocol)
01	001	Name syntax error
02	003	Access control violation
03	003	Abort
04	004	Allocate size too big
05	005	Allocate size overflow
06	006	Improper order for transactions
07	007	Opcode not implemented
08	010	File search failed
09	011	Error described in text message (ASCII characters follow code)

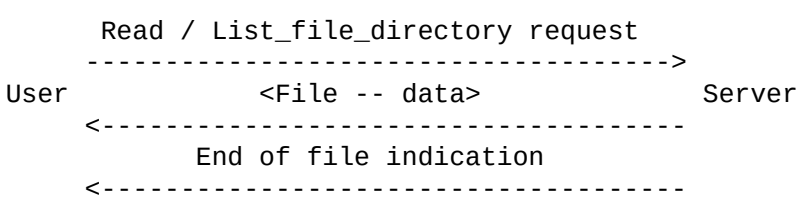
At present, no completion codes are defined for acknowledge. It is assumed that acknowledge refers to the current request being fulfilled.

4. Order of Transactions

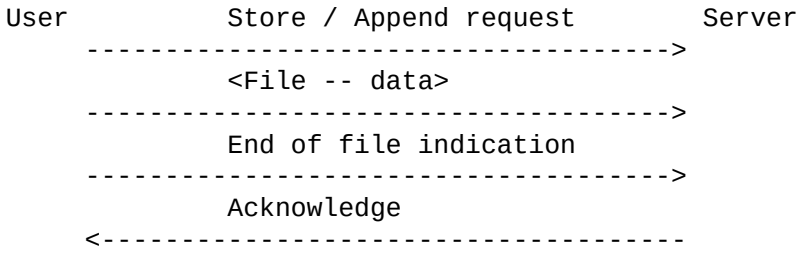
4A. A certain order of transactions must be maintained in fulfilling file transfer requests. The exact sequence in which transactions occur depends on the type of request, as described in section 4B. The fulfilling of a request may be aborted anytime by either host, as explained in section 4C.

4B. Identifier transactions (change data type, username, and password) may be sent by user at any time. The usual order would be a username transaction followed by a password transaction at the start of the connection. No acknowledge is required, or permitted. The identifiers are to be used for default handling, and access control.

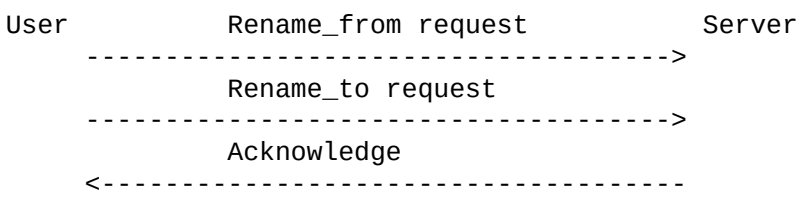
Retrieve and list_file_directory requests cause the transfer of file from server to user. After a complete file has been transferred, the server should indicate end-of-file (by sending CLS or file separator) to complete the request fulfillment sequence, as shown below.



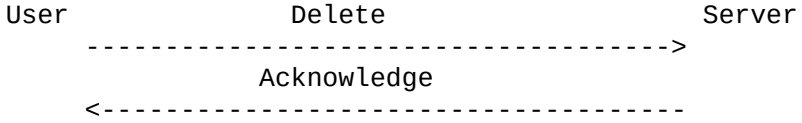
Store and append requests cause the transfer of file from user to server. After a complete file has been transferred, the user should send an end-of-file indication. The receipt of the file must be acknowledged by the server, as shown below.



Rename_from request must be followed by a rename_to request. The request must be acknowledged as shown below.



The delete request requires the server to acknowledge it, as shown below.



Error transactions may be sent by either host at any time, and these terminate the current request fulfillment sequence.

4C. Aborts. Either host may abort a request fulfillment sequence at any time by sending an error terminate, or by closing the connection (NCP to transmit a CLS for the connection). CLS is a more drastic type of abort and shall be used when there is a catastrophic failure, or when abort is desired in the middle of a long transaction. The abort indicates to the receiving host that sender of abort wishes to terminate request fulfillment and is now ready to initiate or fulfill new requests. When CLS is used to abort, the using host will be responsible for reopening connection. The file transfer abort described here is different from the data transfer abort which is sent only by the sender of data. The use of the data transfer abort is not defined in this protocol.

6. Initial Connection, CLS, and Access Control

6A. There will be a preassigned permanent socket number[6] for the cooperating file transfer process at the serving host. The connection establishment will be in accordance with the standard initial connection protocol[7], establishing a full-duplex connection.

6B. The connection will be broken by trading a CLS between the NCP's for each of the two connections. Normally, the user will initiate CLS.

CLS may also be used by either user or server, to abort a transaction in the middle. If CLS is received in the middle of transaction, the current request fulfillment sequence will be aborted. The using host will then reopen connection.

6C. It is recommended that identifier (user name and password) transactions be sent by user to server , at the start, as this would facilitate default handling and access control for the entire duration of connection. The identifier transactions do not require or permit and acknowledge, and the user can proceed directly with requests. If the identifier information is incorrect or not received, the server may send an error transaction indicating access control, violation, upon subsequent requests.

NOTES

[1] Alex McKenzie, BBN, is conducting a survey of network file systems to determine the practicality of standard pathname conventions, and to disseminate information to network users on host file systems.

[2] This initial subset represents control functions necessary for basic file transfer operations, and some elementary file manipulation operations. There is no attempt to provide a data management or complete file management capability.

[3] It is possible that we may, at a later date, assign meaning to these information separators within FTP.

[4] A workable subset is any request, plus terminates. Identifiers may be required in addition when using protected file systems.

[5] It is, however, possible that this bit stream is treated like ASCII characters in specific instances such as transmitting a file to a line printer.

[6] It seems that socket 1 has been assigned to logger. Socket 3 seems a reasonable choice for File Transfer.

[7] RFC 165, or any subsequent standard applicable in initial connection to loggers.