Network Working Group
Request for Comments: 624
NIC #22054
references: RFC 542
obsoletes: RFC 607

Mark Krilanovich (UCSB)
George Gregg (UCSB)
Wayne Hathaway (AMES-67)
Jim White (SRI-ARC)

Feb 1974

Comments on the File Transfer Protocol

This document replaces RFC 607, which was inadvertently released while still in rough draft form. It would be appreciated if RFC 607 were disregarded, and this document considered the accurate statement of the authors' opinions.

There are several aspects of the File Transfer Protocol of RFC 542 that constitute serious drawbacks. Some of these are quite basic in nature, and imply substantial design changes; these will be discussed in a later RFC. Others could be remedied with very little effort, and this should be done as soon as possible.

Following is a list of those problems that can be easily solved, together with their proposed solutions:

1. Once a server has been set to the state where he is "passive" with regard to establishment of data connections, there is no convenient way for the user to make him "active" again. The "REIN" command accomplishes this, but affects more than just the desired active/passive state. SOLUTION: define a new command, with a command verb of "ACTV", to mean that the server is to issue a CONNECT rather than a LISTEN on the data socket. If the server is already "active", the command is a no op. "ACTV" is to have the same reply codes as "PASV".

2. Design of an FTP server or user would be simpler if all command verbs were the same length. While it is certainly possible to handle varying length verbs, fixed length string manipulation is in general easier to write and faster to run than varying length string manipulation, and it would seem that nothing is to be gained in this application by allowing varying length strings. SOLUTION: replace the only three-letter verb, "BYE", with a four-letter one, such as "QUIT", and constrain future command verbs to be four letters long.

3. The order of the handshaking elements following a file transfer command is left unspecified. After sending a STOR command, for example, a user process has no way of knowing which to wait for first, the "250 FILE TRANSFER STARTED" reply, or establishment of the data connection. SOLUTION: specify that the server is to send a "250" reply before attempting to establish the data connection. If it is desired to check if the user is logged in, if the file exists, or if the user is to be allowed access to the file, these checks must be made before any reply is sent. The text of the "250" reply would perhaps be more appropriate as "250 OPENING DATA CONNECTION", since it comes before actual data transfer begins. If the server wishes to send an error reply in the event that the data connection cannot be opened, it is to be sent in lieu of the "252 TRANSFER COMPLETE" reply.

4. Some hosts currently send an error reply on receipt of a command that is unimplemented because it is hot needed (e.g., "ACCT" or "ALLO"). Even though the text of the reply indicates that the command has been ignored, it is obviously impossible for a user process to know that there is no real "error". SOLUTION: require that any server that does not support a particular command because it is not needed in that system must return the success reply for that command.

5. There is no specified maximum length of a TELNET command line, TELNET reply line, user name, password, account, or pathname. It is true that every system implementing an FTP server likely has different maxima for its own parameters, but it is inconvenient, at least in some systems, for the writer of an FTP user (which must converse with many FTP servers) to construct an indefinite length buffer. Similar difficulties confront the writer of a server FTP. SOLUTION: specify a maximum length for TELNET command lines, TELNET replies, user names, passwords, account numbers, and pathnames. This is to be done after conducting a Poll of serving sites concerning their individual maxima. If Network mail is to be included in FTP, the mail text, if sent over the TELNET connection, is to be subject to the same line length maximum.

6. The notion of allowing continuation lines to start with arbitrary text solves a minor problem for a few server FTP implementors at the expense of creating a major problem for all user FTP implementors. The logic needed to decode a multi-line reply is unnecessarily complex, and made an order of magnitude more so by the fact that multi-line replies arc allowed to be nested. SOLUTION: assign a unique (numeric) reply code, such as "009", to be used on all lines of a multi-line reply after the first. The reply code used for this purpose must begin with "0" (it cannot be three blanks, for example), so that it will appear as extraneous to a user process by virtue of the already existing rules concerning reply code groupings.

7. If it is the case that the above solution to (6) is not accepted, the fact that the maximum allowed level of nesting is left unspecified creates a hardship for implementors of user FTPs. This hardship is somewhat easily solved on a machine that has hardware stacks, but not so for other machines. SOLUTION: either disallow nested replies (preferred), or specify a maximum level of nesting of multi-line replies.

8. The prose descriptions of the meanings of the various reply codes are in several cases unclear or ambiguous. For example, the code "020" is explained only as "announcing FTP". It is given as a reply that can be issued when a server cannot accept input immediately after an ICP, but its exact meaning is not obvious. Also. the code "331" is said to mean "ENTER ACCOUNT (if required as part of login sequence)", but is listed as a possible success reply for most of the commands. The explanation indicates that it is only valid in the login sequence, but the command-reply

correspondence table implies that it also means, "I can't do that without an account". SOLUTION: an expanded effort should be made by those who originated the reply codes to define them more completely.

A major complaint about the protocol concerns the fact that the writer of an FTP user process must handle a considerable number of special cases merely to determine Whether or not the last command sent was successful. It is admitted that the protocol is well-defined in all the following areas, but it is important to realize that the characteristic "well-defined" is necessary, but hot sufficient; for many reasons, it is very desirable to employ the simplest mechanism that satisfies all the needs. Following is a list of those drawbacks that unduly complicate the flow chart of an FTP user process:

9. Different commands have different success reply codes. A successful "USER" command, for example, returns a "230", whereas a successful "BYTE" command returns a "200". The stated concept that the first digit would carry this information does not apply, as "100" means success for "STAT", and "200" means success for "SOCK". SOLUTION: specify that any command must return a reply code beginning with some unique digit, such as "2", if successful, and anything other than that digit if not successful. For example this includes changing the success reply for STAT, Perhaps to "200".

10. Some commands have multiple possible success reply codes, e.g., "USER" and "REIN". It is undesirable for ah FTP user to be required to keep a list of reply codes for each command, all of which mean "command accepted, continue". Again, the stated concept concerning the first digit fails, as "230" and "330" are in truth both acknowledgments to a successful "USER" command. SOLUTION: same as for (9) above. The desire to communicate more specific information than simply "yes" or "no", such as the difficulty that some servers do not need all the login parameters, may be solved by having, for example, "230" mean "PASSWORD ACCEPTED, YOU ARE NOW LOGGED IN", and "237" mean "PASSWORD ACCEPTED, ACCOUNT NOW NEEDED". Given the solution to (4) above, a user process becomes much less interested in the difference between "YOU ARE NOW LOGGED IN" and "ACCOUNT NOW NEEDED". The important point is that the idea of "command accepted" is conveyed by the initial "2, and that finer gradations of meaning can be deduced by the user process, if desired.

11. The meanings of the various connection greeting reply codes are somewhat inconsistent. "300 connection greeting, awaiting input", if intended as a positive acknowledgments to the ICP, should be a 200-series reply, or if intended to be purely informative, a 000-series reply. If the former, then clearly "020 expected delay" is the corresponding negative acknowledgments, and should be a 400-series reply. It is however unlikely that notification of an expected delay would be of importance to a user Process without knowledge of the length of the delay. SOLUTION.: change "300 connection greeting" to a 000-series reply, perhaps

"011" (preferred), or change "300 connection greeting" to a 200-series reply, perhaps "211", and "020 expected delay" to a 400-series reply, perhaps "411".

In addition to the above mentioned weaknesses in the protocol, the following is believed to be a typographical error:

12. Reply code "332 LOGIN PLEASE" is not listed anywhere in the command-reply correspondence table. It Would seem that this would be a more-information-needed (success) reply for all those commands which require the user to be logged in. It should also be stressed that the "332" code is to be used for this purpose, as many servers currently use other codes, such as "451" and "504", to mean "LOGIN PLEASE".