This document describes a revised set of reply codes for the File Transfer Protocol.

The aim of this revision is to satisfy the goal of using reply codes to enable the command issuing process to easily determine the outcome of each command. The user protocol interpreter should be able to determine the success or failure of a command by examining the first digit of the reply code.

An important change in the sequencing of commands and replies which may not be obvious in the following documents concerns the establishment of the data connection.

In the previous FTP specifications when an actual transfer command (STOR, RETR, APPE, LIST, NLIST, MLFL) was issued the preliminary reply was sent after the data connection was established. This presented a problem for some user protocol interpreters which had difficulty monitoring two connections asynchronously.

The current specification is that the preliminary reply to the actual transfer commands indicates that the file can be transferred and either the connection was previously established or an attempt is about to be made to establish the data connection.

This reply code revision is a modification of the protocol in described in RFC 542, that is to say that the protocol implementation associated with socket number 21 (decimal) is the protocol specified by the combination of RFC 542 and this RFC.

A note of thanks to those who contributed to this work: Ken Pogran, Mark Krilanovich, Wayne Hathway, and especially Nancy Neigus.
Replies to File Transfer Protocol commands were devised to ensure the synchronization of requests and actions in the process of file transfer, and to guarantee that the user process always knows the state of the Server. Every command must generate at least one reply, although there may be more than one; in the latter case, the multiple replies must be easily distinguished. In addition, some commands occur in sequential groups, such as USER, PASS and ACCT, or RNFR and RNTO. The replies show the existence of an intermediate state if all preceding commands have been successful. A failure at any point in the sequence necessitates the repetition of the entire sequence from the beginning.

Details of the command-reply sequence will be made explicit in a state diagram.

An FTP reply consists of a three digit number (transmitted as three alphanumeric characters) followed by some text. The number is intended for use by automata to determine what state to enter next; the text is intended for the human user. It is intended that the three digits contain enough encoded information that the user-process (the User-PI described in RFC 542) will not need to examine the text and may either discard it or pass it on to the user, as appropriate. In particular, the text may be server-dependent, so there are likely to be varying texts for each reply code.

Formally, a reply is defined to contain the 3-digit code, followed by Space <SP>, followed by one line of text (where some maximum line length has been specified), and terminated by the TELNET end-of-line code. There will be cases, however, where the text is longer than a single line. In these cases the complete text must be bracketed so the User-process knows when it may stop reading the reply (i.e. stop processing input on the TELNET connection) and go do other things. This requires a special format on the first line to indicate that more than one line is coming, and another on the last line to designate it as the last. At least one of these must contain the appropriate reply code to
indicate the state of the transaction. To satisfy all factions it was decided that both the first and last line codes should be the same.

Thus the format for multi-line replies is that the first line will begin with the exact required reply code, followed immediately by a Hyphen, "-" (also known as Minus), followed by text. The last line will begin with the same code, followed immediately by Space <SP>, optionally some text, and TELNET <eol>.

For example:

```
123-First line
Second line
  234 A line beginning with numbers
 123 The last line
```

The user-process then simply needs to search for the second occurrence of the same reply code, followed by <SP> (Space), at the beginning of a line, and ignore all intermediary lines. If an intermediary line begins with a 3-digit number, the Server must pad the front to avoid confusion.

This scheme allows standard system routines to be used for reply information (such as for the STAT reply), with "artificial" first and last lines tacked on. In the rare cases where these routines are able to generate three digits and a Space at the beginning of any line, the beginning of each text line should be offset by some neutral text, like Space.

This scheme assumes that multi-line replies may not be nested. We have found that, in general, nesting of replies will not occur, except for random system messages (called spontaneous replies in the previous FTP incarnations) which may interrupt another reply. Spontaneous replies are no longer defined; system messages (i.e. those not processed by the FTP server) will NOT carry reply codes and may occur anywhere in the command-reply sequence. They may be ignored by the User-process as they are only information for the human user.

The three digits of the reply each have a special significance. This is intended to allow a range of very simple to very sophisticated response by the user-process. The first digit denotes whether the response is good, bad or incomplete. (Referring to the state diagram) an unsophisticated user-process will be able to determine its next action (proceed as planned, redo, retrench, etc.) by simply examining this first digit. A user-process that wants to know approximately what kind of error
occurred (e.g. file system error, command syntax error) may examine the second digit, reserving the third digit for the finest gradation of information (e.g. RNTO command without a preceding RNFR.)

There are four values for the first digit of the reply code:

1yz  Positive Preliminary reply

The requested action is being initiated; expect another reply before proceeding with a new command. (The user-process sending another command before the completion reply would be in violation of protocol; but server-FTP processes should queue any commands that arrive while a preceeding command is in progress.) This type of reply can be used to indicate that the command was accepted and the user-process may now pay attention to the data connections, for implementations where simultaneous monitoring is difficult.

2yz  Positive Completion reply

The requested action has been successfully completed. A new request may be initiated.

3yz  Positive Intermediate reply

The command has been accepted, but the requested action is being held in abeyance, pending receipt of further information. The user should send another command specifying this information. This reply is used in command sequence groups.

4yz  Transient Negative Completion reply

The command was not accepted and the requested action did not take place, but the error condition is temporary and the action may be requested again. The user should return to the beginning of the command sequence, if any. It is difficult to assign a meaning to "transient", particularly when two distinct sites (Server and User-processes) have to agree on the interpretation. Each reply in the 4yz category might have a slightly different time value, but the intent is that the user-process is encouraged to try again. A rule of thumb in determining if a reply fits into the 4yz or the 5yz (Permanent Negative) category is that replies are 4yz if the commands can be repeated without any change in command form or in properties of the User or Server (e.g. the command is spelled the same with the same
arguments used; the user does not change his file access or
user name; the server does not put up a new
implementation.)

5yz Permanent Negative Completion reply

The command was not accepted and the requested action did
not take place. The User-process is discouraged from
repeating the exact request (in the same sequence). Even
some "permanent" error conditions can be corrected, so the
human user may want to direct his User-process to
reinitiate the command sequence by direct action at some
point in the future (e.g. after the spelling has been
changed, or the user has altered his directory status.)

The following function groupings are encoded in the second
digit:

x0z Syntax - These replies refer to syntax errors,
syntactically correct commands that don’t fit any
functional category, unimplemented or superfluous
commands.

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

x2z Connections - Replies referring to the TELNET and
data connections.

x3z Authentication and accounting - Replies for the logon
process and accounting procedures.

x4z Unspecified as yet

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

The third digit gives a finer gradation of meaning in each of
the function categories, specified by the second digit. The
list of replies below will illustrate this. Note that the
text associated with each reply is suggestive, rather than
mandatory, and may even change according to the command with
which it is associated. The reply codes, on the other hand,
should strictly follow the specifications in the last section;
that is, Server implementations should not invent new codes
for situations that are only slightly different from the ones
described here, but rather should adapt codes already defined.
If additional codes are found to be necessary, the details should be submitted to the FTP committee, through Jon Postel.

A command such as TYPE or ALLO whose successful execution does not offer the user-process any new information will cause a 200 reply to be returned. If the command is not implemented by a particular Server-FTP process because it has no relevance to that computer system, for example ALLO at a TENEX site, a Positive Completion reply is still desired so that the simple User-process knows it can proceed with its course of action. A 202 reply is used in this case with, for example, the reply text: "No storage allocation necessary." If, on the other hand, the command requests a non-site-specific action and is unimplemented, the response is 502. A refinement of that is the 504 reply for a command that IS implemented, but that requests an unimplemented parameter.

200 Command okay

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

202 Command not implemented, superfluous at this site.

501 Syntax error in parameters or arguments

204 Command not implemented for that parameter

202 Command not implemented, superfluous at this site.

504 Command not implemented for that parameter

110 Restart marker reply.
   In this case the text is exact and not left to the
   particular implementation; it must read:
   MARK yyyy = mmmm
   where yyyy is User-process data stream marker, and
   mmmm is Server’s equivalent marker. (note the
   spaces between the markers and ".")

211 System status, or system help reply

212 Directory status

213 File status

214 Help message (on how to use the server or the meaning
   of a particular non-standard command. This reply
   is useful only to the human user.)

120 Service ready in nnn minutes

220 Service ready for new user

221 Service closing TELNET connection (logged off if
   appropriate)

421 Service not available, closing TELNET connection.
   [This may be a reply to any command if the service
   knows it must shut down.]
125  Data connection already open; transfer starting  
225  Data connection open; no transfer in progress  
425  Can't open data connection  
226  Closing data connection; requested file action  
    successful (for example, file transfer or file  
    abort.)  
426  Connection trouble, closed; transfer aborted.  
227  Entering [passive, active] mode  
230  User logged on, proceed  
530  Not logged in  
331  User name okay, need password  
332  Need account for login  
532  Need account for storing files  
150  File status okay; about to open data connection.  
250  Requested file action okay, completed.  
350  Requested file action pending further information  
450  Requested file action not taken: file unavailable  
    (e.g. file not found, no access)  
550  Requested action not taken: file unavailable (e.g.  
    file busy)  
451  Requested action aborted: local error in processing  
452  Requested action not taken: insufficient storage  
    space in system  
552  Requested file action aborted: exceeded storage  
    allocation (for current directory or dataset)  
553  Requested action not taken: file name not allowed  
354  Start mail input; end with <CR><LF>.<CR><LF>  

In this section, the command-reply sequence is presented. Each  
command is listed with its possible replies; command groups are  
listed together. Preliminary replies are listed first (with  
their succeeding replies under them), then positive and negative  
completion, and finally intermediary replies with the remaining  
commands from the sequence following. This listing forms the  
basis for the state diagrams, which will be presented separately.  

ICP  
120  
220  
220  
421  

Command-Reply Sequences
Logon

USER
230
530
500, 501, 421
331, 332

PASS
230
202
530
500, 501, 503, 421
332

ACCT
230
202
530
500, 501, 503, 421

Logoff

QUIT
221
500

REIN
120
220
220
421
500, 502

Transfer parameters

SOCK
200
500, 501, 421, 530

PASV
227
500, 501, 502, 421, 530

ACTV
227
202
500, 501, 421, 530

BYTE, MODE, TYPE, STRU
200
500, 501, 504, 421, 530
File action commands

ALLO
  200
  202
  500, 501, 504, 421, 530

REST
  500, 501, 502, 421, 530
  350

STOR
  125, 150
    (110)
  226, 250
  425, 426, 451, 552
  532, 450, 452, 553
  500, 501, 421, 530

RETR
  125, 150
    (110)
  226, 250
  425, 426, 451
  450, 550
  500, 501, 421, 530

LIST, NLST
  125, 150
  226, 250
  425, 426, 451
  450
  500, 501, 502, 421, 530

APPE
  125, 150
    (110)
  226, 250
  425, 426, 451, 552
  532, 450, 550, 452, 553
  500, 501, 502, 421, 530

MLFL
  125, 150
  226, 250
  425, 426, 451, 552
  532, 450, 550, 452, 553
  500, 501, 502, 421, 530

RNFR
  450, 550
  500, 501, 502, 421, 530
  350

RNTO
  250
  532, 553
Neigus                                      FTP Reply Codes  [10]

500, 501, 502, 503, 421, 530  13e9c
DELE  13e10
   250  13e10a
   450, 550  13e10b
   500, 501, 502, 421, 530  13e10c
ABOR  13e11
   225, 226  13e11a
   500, 501, 502, 421  13e11b
MAIL  13e12
   354  13e12a
       250  13e12a1
       451, 552  13e12a2
   450, 550, 452, 553  13e12b
   500, 501, 502, 421, 530  13e12c

Informational commands  13f

STAT  13f1
   211, 212, 213  13f1a
   450  13f1b
   500, 501, 502, 421, 530  13f1c
HELP  13f2
   211, 214  13f2a
   500, 501, 502, 421  13f2b

Miscellaneous commands  13g

SITE  13g1
   200  13g1a
   202  13g1b
   500, 501, 530  13g1c
NOOP  13g2
   200  13g2a
   500  13g2b
Here we present state diagrams for a very simple minded FTP implementation. Only the first digit of the reply codes is used. There is one state diagram for each group of FTP commands or command sequences.

The command groupings were determined by constructing a model for each command then collecting together the commands with structurally identical models.

For each command or command sequence there are three possible outcomes: success (S), failure (F), and error (E). In the state diagrams below we use the symbol B for "begin", and the symbol W for "wait for reply".

We first present the diagram that represents the largest group of FTP commands:

```
+---+    cmd    +---+    2      +---+
! B !---------->! W !---------->! S !
+---+           +---+           +---+
+---+    4,5    +---+
!        !---------->! F !
+---+
```

This diagram models the commands:

ABOR, ACTV, ALLO, BYTE, DELE, HELP, MODE, NOOP, PASV, QUIT, SITE, SOCK, STAT, STRU, TYPE.
The other large group of commands is represented by a very similar diagram:

```
+---+ 3 +----+
    +-----+----+
    ! B ! 1,2 2 +----+
    ! E ! ! ! !
    +----+
```

This diagram models the commands:

APPE, (ICP), LIST, MLFL, NLST, REIN, RETR, STOR.

Note that this second model could also be used to represent the first group of commands, the only difference being that in the first group the 100 series replies are unexpected and therefore treated as error, while the second group expects (some may require) 100 series replies.

The remaining diagrams model command sequences, perhaps the simplest of these is the rename sequence:

```
+----+ RNFR 1,2 +----+
+----+ ! B !----+
     +----+
     ! E !
     +----+
     ! 1,2 !
```

21a
A very similar diagram models the Mail command:

```
+----+ MAIL +----+ 1,2 +----+
! B !<-------->! W !<-------->! E !
+----+        +----+        -->+----+
        3 ! ! 4,5 !
---------- ------ !
        2 ! ! 1,3 ! ! +----+
        ! ! ! !
V ! ! ! !
+----+ text +----+ 4,5 ------+----+
! !<-------->! W !<-------->! F !
+----+ +----+ +----+
```

Note that the "text" here is a series of lines sent from the user to the server with no response expected until the last line is sent, recall that the last line must consist only of a single period.
The next diagram is a simple model of the Restart command:

```
+----+    REST    +----+  1,2    +----+
! B !------------->! W !----------->! E !
+----+        --+----+        -->+----+
            3   ! ! 4,5   !
------------  ------   
! ! 2! ------
! ! ! !
V   ! ! !
+----+    cmd    +----+  4,5 ----->+----+
! !---------->! W !----------->! F !
+----+        -->+----+        
    ! !
    ! 1 !
-----
```

Where "cmd" is APPE, STOR, RETR, or MLFL.

We note that the above three models are similar, in fact the Mail diagram and the Rename diagram are structurally identical. The Restart differs from the other two only in the treatment of 100 series replies at the second stage.
The most complicated diagram is for the Logon sequence:

```
1
++-+ USER +----------->++-+
! B !--------->! W ! 2 ------->! E !
++-+ +--------- ! -->++-+
      ! ! ! ! !
3 ! ! 4,5 ! ! !
-------------- ---- ! !
! ! ! ! ! !
! ! ! ! ! !
! ! ! ! ! !
! ! ! ! ! !
! ! ! ! ! !
V ! ! ! ! ! !
++-+ PASS ++-+ 2 ! ------->++-+
! !---------->! W !----------->! S !
++-+ ++-+ ---------->++-+
      ! ! ! ! ! !
3 ! ! 4,5 ! ! !
-------------- ------ !
! ! ! ! ! !
! ! ! ! ! !
! ! ! ! ! !
! ! ! ! ! !
! ! ! ! ! !
V ! ! ! ! ! !
++-+ ACCT ++-+ ! ------->++-+
! !---------->! W ! 4,5 ------->! F !
++-+ +----------->++-+
```

25a
Finally we present a generalized diagram that could be used to model the command and reply interchange:

```
Begin

+-----+ cmd +-----+ 2 +-----+ 2
---! !-------! !-------->! ! !
---! ! ! W ! ! S !-----!
---! ! !>! !------ ! ! !
+-----+ ! +----+ 4,5 ! +----+ !
---! ! ! ! !!! !
---! ! ! ! 1!3 ! +----+!
---! ! ! ! !!! ! ! ! !
---! ! ---- ! ---->! F !----
---! ! ! ! !
---! ! ! +----
```

End

26a