Postel
12762

Rucd 21Dec72

NETWORK WORKING GROUP
Request for Comments #418
NIC #12762
Categories D.7, G.3

Wayne Hathaway (AMES-67)
November 27, 1972

A414

Server File Transfer Under TSS/360
At NASA-Ames Research Center

This RFC is a description of the initial implementation of Server File Transfer at NASA-Ames Research Center. It also contains some comments on the current File Transfer Protocol (RFC's 354 and 385) and some suggestions for future extensions. Comments or questions should be directed to:

Wayne Hathaway
Mail Stop 233-9
NASA-Ames Research Center
Moffett Field, CA 94035
(415) 965-6033 or
(408) 736-7439

TSS FILE MANAGEMENT

The ARPA Network file transfer facilities provided at Ames interface very closely with the standard TSS file management facility; network users should be familiar with TSS in general and its file handling capabilities in particular. A few of the basic concepts are mentioned here. Each person allowed to use TSS is assigned a user identification (USERID), which is padded to eight characters with asterisks by the system. At Ames, the userid is normally formed from a three letter organization code and the user's initials. Each userid has its own catalog (directory), which contains pointers to all datasets (files) which may be accessed by the user. TSS dataset names are made up of one or more qualification levels, up to eight characters each, separated by periods. The highest qualification level is in fact the userid, but this is not entered when specifying dataset names; the currently active userid is appended by the system to all names entered by the user. The levels of qualification form a tree structure which may be used to identify logical groups of datasets. Only ends of branches actually name datasets; such names are termed "fully qualified" names. Partially qualified names are used in many TSS dataset management commands but are not used in ARPA Network File Transfer. Most datasets under TSS are contained on a pool of universally available disks termed "public storage." Datasets on public storage are automatically cataloged at creation. Datasets may also exist on private devices, either disks or tapes. Private datasets may be cataloged or not, as the user desires. Only disk datasets in the "virtual access method" (VAM) are supported by TSS Server File Transfer, but private VAM datasets may be used if cataloged. Ample TSS commands exist for transferring VAM datasets to tape, cards, printer, etc.

The TSS catalog structure also allows extensive sharing of datasets among users. This sharing may be done at any qualification level and with read-only, read-write, or unlimited access (only unlimited sharing allows erasing). To share a TSS dataset, the owner must first "permit" it to one or more users. Each user must then explicitly "share" it, which causes an entry for the dataset to be made in his catalog. This entry, which may be under a different name than the original, does not point to the dataset itself, but rather points to the appropriate entry in the owner's catalog. This allows the owner to change or remove sharing access at any time.

TSS FILE STRUCTURE

A brief discussion is also in order about the difference between unstructured files and record-structured files. All files on TSS are stored with some form of record structure, although each record may be up to one million bytes in length; the important thing is the meaning attributed to a record.

For character oriented datasets (e.g., source programs, listings, program documentation) a record corresponds to a line; there are no end-of-record markers in the data itself (CR-LF, NL, EOL). The TSS editors require this, the translators require this and so forth. This does not preclude having an EBCDIC "new-line" character (NL) in the data, how-

ever; it is just treated as part of the line of data. Thus character oriented files must be sent to and stored on TSS with record structuring to be useful.

The use of unstructured files on TSS is usually restricted to binary data and object programs. From the network the only use for such files would be data which is in fact a totally unstructured string of bits (using IMAGE type). An example might be an object program for an IMLAC.

Because of the above, our server has implemented the following conventions.

Sending structured data (RETR, STRU R):

> An end-of-record sequence will be added to each record read from the TSS dataset (either "stream" mode CR-LF or "text" mode EOR).

Sending unstructured data (RETR, STRU F):

> The data will be sent with no end-of-record sequence added.

Receiving structured data (STOR or APPE, STRU R):

> An end-of-record sequence is required to define each TSS record (with a maximum record length of 1012 bytes).

Receiving unstructured data (STOR or APPE, STRU F):

> Every character received is written into arbitrary sized records (except for "text" mode EOF character).

The above implementation does satisfy one criterion (as stated in the third sentence of III.C in RFC 354): any file sent to TSS may be retrieved exactly as sent. For example, this means that if a character file is sent in unstructured form, the CR-LF sequences will be stored as part of the data. When this dataset is retrieved (also in unstructured form) no new end-of-record indicators will be added, so the retrieved file will have only the original CR-LF record information. If the file were retrieved in record-structured form, however, this would of course not be the case; it is important that any unstructured file be retrievedin exactly the same manner that it was sent.

This criterion of retrievability is not the main one of concern to TSS users, however; a more important criterion is that files sent to TSS may be of use on that system (i.e., may be edited, may be printed, etc.). To accomplish this it is imperative that character data be sent with record structure.

As can be seen from the above discussion, this should not be a problem with most hosts, as in default transfers (MODE S, TYPE A) they are already including end-of-record information at the end of each physical line (CR-LF). Thus the only requirement would be to not reject the STRU R command (at least in MODE S, TYPE A transfers). This may cause

some minor problems (such as CR-LF-LF-LF-LF for vertical spacing) but it is strongly requested that hosts which do not usually store files in record-structured form at least implement STRU R in this simple context.

TSS SERVER FILE TRANSFER IMPLEMENTATION

The TSS implementation of each of the file transfer commands is described below. The maximum length for a command and its parameters is 256 characters (not including the CR-LF at the end). All blanks are removed from the parameter string, and so blanks are never significant except to separate a command from its parameters.

Only the commands MAIL and MLFL may be used when there is no active user logged on; all other commands require a prior USER command.

USER

The USER command defines the TSS userid to be employed for catalog access. This must be a valid TSS userid, but it must also have been specified as having access to network file transfer. This is done by installation management in a manner similar to establishing a valid TSS userid; contact Wayne Hathaway to have existing TSS userids joined to ARPA Network File Transfer. The demonstration userids ARPA, ARPA1, and ARPA2 do have access to file transfer.

The USER command also resets all file transfer parameters to their default values, including HOST and SOCK. It is felt this should have been specified in the protocol.

If a USER command is entered during a file transfer operation it will be held until the transfer is complete; the new userid will then be processed, a password may be requested, etc.

PASS

TSS file transfer passwords are optional and are established when userids are granted file transfer access. This password is not necessarily the same as the normal TSS password. The demonstration userids ARPA, ARPA1, and ARPA2 have no passwords.

ACCT

The ACCT command is not supported.

The protocol seems somewhat deficient in the definition of the ACCT command, particularly in the assigning of reply codes. I would like to propose that code 331 be assigned to "ENTER ACCT" when the ACCT command is required for "log-on" and code 433 be assigned to "ENTER ACCT" when the ACCT command is required only for a particular transfer operation. It is felt that this would aid user automata in satisfying particular server requirements.

BYTE

The only BYTE size supported is eight.

SOCK

The SOCK command is supported in accordance with the protocol. There is only one HOST parameter, however, which applies to both send and receive sockets.

TYPE

Types currently supported are ASCII and IMAGE. The EBCDIC type will be added later. The two "print file" types are not yet supported, due partly to lack of time and partly to a seeming fault in the protocol: it appears the only transfer mode which allows a "print file" is BLOCK. This is based on the descriptions of transfer modes on page 12 of RFC 354: TEXT mode allows only ASCII type, and STREAM mode allows record structures (surely necessary in print files) only in ASCII type. I assume this is an oversight, and we in fact intend to support EBCDIC type in STREAM mode with record structures (since EBCDIC contains both CR and LF). If not, the implementation of "print file" types will be delayed substantially.

STRU

Both file and record structures are supported, in accordance with the protocol. The EBCDIC type will be supported with record structure, however, as indicated under the TYPE command.

MODE

The STREAM and TEXT modes are currently supported for the indicated types (STREAM is required for IMAGE type). In STREAM mode with record strutures, the CR-LF sequence (with optional Telnet NOP's) is required as EOR. In TEXT mode with no record structure any EOR characters encountered are discarded. In TEXT mode with record structure a message containing only an EOF character is assumed to be an end-of-file only, not a null record with an omitted EOR (although we would like to continue our lobbying against "implied EOR" concepts!).

RETR

The RETR command is supported in accordance with the protocol; the parameter must be a fully qualified name of an existing dataset on direct access storage (public or private). For structured files in ASCII type a "CR-LF" sequence is inserted at the end of each record. No EOR characters are inserted for unstructured files in TEXT mode.

STOR

The STOR command is supported in accordance with the protocol; the parameter must be a valid fully qualified dataset name, with unlimited or read-write access required for existing shared datasets. The characteristics of the TSS dataset being stored are determined as follows.

For new datasets, the default characteristics are VS organization with a maximum record size of 1012 bytes. This may be overridden by specifying an ALLO (TSS "DDEF") command with different characteristics. A TSS "line dataset" may thus be created by first entering an ALLO command specifying VI organization (line numbers will be inserted by the system). For other types of VI datasets, however, the user must ensure that the records contain keys in the specified positions and that they are sent in order by key; it is anticipated that this facility will be rarely used.

For existing datasets, the new version will be created with the same characteristics as the original (and on the same private volume).

APPE

The APPE command is supported in accordance with the protocol; the parameter must be a valid fully qualified dataset name, with unlimited or read-write access required for existing shared datasets. The characteristics of created datasets are determined as for the STOR command.

RNFR

The RNFR command is supported in accordance with the protocol; the parameter must specify the fully qualified name of an existing dataset.

RNTO

The RNTO command is supported in accordance with the protocol; the parameter must specify the fully qualified name of a nonexistent dataset.

DELE

Due to the TSS file sharing and private device facilities, the single DELE command is not really adequate. In TSS there are two separate commands for this function, ERASE and DELETE. The ERASE command is used to actually purge a dataset and free its space; it may be used only on direct access datasets (public or private) with unlimited access. The DELETE command simply removes a catalog entry and does not touch the dataset itself; it may be used on shared datasets of any type (to effectively "unshare") or on datasets on private devices. To attempt to provide both of these functions, the DELE command has been implemented as follows.

For nonshared datasets:

Public or private direct-access:
erase (free the space) and delete the catalog entry

Private nondirect-access:
delete the catalog entry only

For shared datasets:

Public or private direct-access, unlimited sharing:
erase (free the space) and delete the catalog entry.

All others:
delete the sharer's catalog entry only

ALLO

The ALLO command is used to enter a TSS "DDEF" command. This may be used to specify nonstandard dataset attributes, private devices, or other items. The parameter is a complete TSS "DDEF" command minus the word "DDEF" itself, with the minor restriction that the first three parameters (DDNAME, DSORG, and DSNAME) must be present and must not be specified in keyword format.

REST

The REST command is not supported. It will be added when BLOCK mode is implemented.

STAT

There are three uses of the STAT command. If used between file transfers with no parameter, it will print the current values for all file transfer parameters (user, byte size, host, receive data socket, send data socket, transfer type, file structure, and transfer mode). If used during a file transfer operation (with or without a parameter) it will indicate what type of operation is active.

The third use of STAT is to retrieve status information about a file or group of files (i.e., any qualification level). There are two forms for this information, termed short and long. The short form will indicate what datasets exist with the qualification level specified, with information about sharing and private devices. If the parameter specified is a fully qualified name, the entire dataset name is printed. If it is partially qualified, the names of all datasets in that group are printed (with only the unique portion of the names actually appearing). This short form is exactly equivalent to the TSS "PC?" command and is the default form for STAT.

The parameter for the long form is the same as for the short form, but all available information is printed for each dataset (at least five lines per dataset). To get the long form, the characters ",LONG" must be appended to the parameter. The long form is exactly equivalent to the TSS "DSS?" command.

The STAT command does not currently send a reply code 200 at all. It is felt this should be added to the protocol to better define the end of STAT information; the reply code 200 will be added at that time.

LIST

The LIST command produces output identical to the short and long form STAT command (parameter required) except that the long form is the default; to get the short form, the characters ",SHORT" must be appended to the parameter.

ABOR

The ABOR command may be entered at any time during a file transfer operation. It will cause an immediate termination of the transfer and closing of the data connection. For RETR operations, the TSS dataset will be unaffected. For STOR or APPE on a new dataset, there will be nothing created (except possibly an intermediate file which should be ignored). For STOR of an existing dataset, the dataset will be unchanged. For APPE to an existing dataset, all transferred information will have been written into the dataset.

BYE

The BYE command is supported in accordance with the protocol. If a BYE command is entered during a transfer operation the next command entered (which must be a USER command) will not be processed at all until the transfer is complete. Note that this precludes aborting the transfer after a BYE command is entered (which seems to fit the spirit of BYE).

MAIL

The MAIL command will accept a TSS userid, a known NIC ident (i.e., one for which a mapping to TSS userid exists), or a null parameter (in which case the mail will be sent to the TSS systems programming group). Unknown idents will cause the MAIL command to be ignored.

The MAIL command may be entered prior to the first USER command, to increase the utility of the network mail system. This is not currently mentioned in the protocol; it is felt that this implementation should be recommended.

MLFL

The parameter is the same as for the MAIL command.

The MLFL command may also be entered prior to a USER command, for the same reasons indicated under MAIL.

COMMENTS ON THE PROTOCOL

Some possible additions to the File Transfer Command Set are as follows.

NOP

It seems strange to find any protocol without a no-operation command, and there are several possible uses for such a command. Assuming there is a special reply code assigned to "NOP COMMAND ACCEPTED," then NOP could serve the same function as the level 2 ECO (for user verification of server existence, for example). Another example could be in allowing the user to discard terminal output, as from a too-lenghty STAT command. The problem with just discarding everything is that the user process does not know when all output has been received. If a NOP command were defined, the user process could send a NOP upon receipt of the discard request, and then discard until the unique NOP response is received.

SRVR

The SRVR command would be used to pass a parameter which has significance only to the particular server system. This parameter might specify printing a newly created file, moving a file, or any number of server-defined functions. One possible implementation would be for the parameter to be a valid server system command, which would then be "obeyed" by the server.

To simplify implementation of both SRVR and STAT commands, the following change to the protocol is requested. Between the receipt of a STAT or SRVR command and the sending of a recognizable reply code (i.e., any message with numbers in the first three positions), a server is allowed to send messages which do not have valid reply codes. Thus the server system response messages could be sent with no modification (except to insure that the first three positions are not numeric). These messages would be assumed to have code 000 and should be printed by any user processes. This technique could possibly be extended to other commands with good results.

CLSE

During implementation of the File Transfer Server the following operating mode was considered, that of the so-called "hot card reader." In this mode a connection would be established between two systems once to be left open for an extended period. Individual users would enter USER commands, transfer files, and then enter BYE commands to safeguard their files. For this mode the BYE command should not close the TELNET connection. In fact, this is the definition of the BYE command if entered during a transfer and if a new USER command comes before the transfer is complete.

To eliminate the timing uncertaintities of the BYE command, and also to allow the "hot card reader" mode, it is recommended that the BYE command be redefined to mean "terminate the currently active user and require a new USER command before additional transfers." The TELNET connections could of course be closed by the user process as soon as the BYE command is acknowledged. For user processes that cannot or do not choose to do this, a new command CLSE should be added, to have the meaning of "close the TELNET connection when possible (i.e., after any active transfer is complete)."

Thus BYE would have only the meaning it currently does if entered during an active transfer, and CLSE would have the current meaning of BYE in other circumstances. Further suggestions would be that CLSE implies a BYE and that a CLSE during an active transfer "logically" closes the connection immediately (i.e., no commands may be entered while waiting for the transfer to complete).

Some small items missing from the protocol are now mentioned.

DEFAULT PARAMETER VALUES

It should be more emphatically stated that servers must assume default values for all unspecified parameters (MODE, TYPE, etc.). Without this, all user processes must always send a complete set of parameter specification commands, since some server somewhere may not make the correct default assumptions. This is obviously undesirable.

SYNCH ON THE TELNET CONNECTION

The protocol states that the ABOR command should be preceded by the Telnet SYNCH condition, for reasons of efficiency (so that servers do not need to be continually checking the Telnet connection during a transfer). This should be extended to other commands which may be used during a transfer (BYE, USER, STAT, etc.).

BREAK ON THE TELNET CONNECTION

The question of what a BREAK character means over the Telnet connection is not touched. Even if the BREAK is to have no function this should be specified. In the TSS implementation, the only time a BREAK is recognized is during a MAIL command; the BREAK will abort the command and cancel the mail.

CATCH-ALL REPLY CODES

It is felt that one reply code of each type should be assigned the meaning of "other" so that system-dependent events could be properly identified. For example, if an indexed file is being sent to TSS and records are received out of order, the reply code currently used is 453, although this is supposed to mean "insufficient space." If a catch-all code were defined, the message text could be used to further specify the exact meaning.

PARAMETER VALIDATION

Certain server processes (notably TENEX) have apparently divided the file transfer commands into two logical groups: parameter setting commands and file manipulation commands. The parameter setting commands are TYPE, MODE, STRU, and BYTE. The parameter values specified on these commands are not validated when received; they are simply stored and a reply code 200 is returned. The parameters are validated on the next file manipulation command, however, with errors resulting in cancellation of the file manipulation command itself with reply code 503.

I am aware of the problems with validating parameters when received (i.e., of temporarily producing invalid combinations when changing from one valid set to another) but it is felt that some standard could be set up to allow this. As an absolute minimum, totally invalid or unsupported parameter values should be diagnosed immediately (e.g., MODE B if BLOCK mode is not implemented).

This problem is a nuisance with a human user, but it becomes quite serious with a user automaton: the amount of work necessary for an automaton to empirically determine a valid set of parameters is intolerable.