FTP Reference Manual

Table of Contents

c Xerox Corporation 1979

1. Introduction

FTP is a Pup-based File Transfer Program for moving files to and from an Alto file system. The program has three main parts:

1. An FTP Server, which listens for file transfer requests from other hosts,
2. An FTP User, which initiates file transfers under control of either the keyboard or the command line, and
3. A User Telnet for logging into a remote host using the Pup Telnet protocol.

1.1 Concepts and terminology

Tranferring a file from one machine (or *host*) to another over a network requires the active cooperation of programs on both machines. In a typical scenario for file transfer, a human user (or a program acting on his behalf) invokes a program called an *FTP User* and directs it to establish contact with an *FTP Server* program on another machine. Once contact has been established, the FTP User initiates requests and supplies parameters for the actual transfer of files, which the User and Server proceed to carry out cooperatively. The FTP User and FTP Server roles differ in that the FTP User interacts with the human user (usually through some sort of keyboard interpreter) and takes the initiative in user/server interactions, whereas the FTP Server plays a comparatively passive role.

The question of which machine is the FTP User and which is the FTP Server is entirely independent of the direction of file transfer. The two basic file transfer operations are called *Retrieve* and *Store*; the Retrieve operation causes a file to move from Server to User, whereas Store causes a file to move from User to Server.

The Alto FTP subsystem contains both an FTP User and an FTP Server, running as independent processes. Therefore, to transfer files between a pair of Altos, one should start up the FTP subsystem on both machines, then issue commands to the FTP User process on one machine directing it to establish contact with the FTP Server process in the other machine. Subsequent file transfers are controlled entirely from the FTP User end, with no human intervention required at the Server machine.

Transferring files to or from a Maxc system or an IFS involves establishing contact with FTP Server processes that run all the time on those machines. Hence, one may simply invoke the Alto FTP subsystem and direct its FTP User process to connect to the machine.

In the descriptions that follow, the terms *local* and *remote* are relative to the machine on which the FTP User program is active. That is, we speak of typing commands to our local FTP User program and directing it to establish contact with an FTP Server on some remote machine. A Retrieve command then copies a file from the remote file system to the local file system, whereas a Store command copies a file from the local file system to the remote file system.

Furthermore, we refer to *local* and *remote filenames*. These must conform to the conventions used by the local and remote host computers, which may be dissimilar (for example, Alto versus Maxc). The Alto FTP knows nothing about Maxc filename conventions or vice versa.

The Alto FTP subsystem also includes a third process, called a *User Telnet*, which simulates a terminal in a manner exactly analogous to the Chat subsystem (though lacking some of its finer features). By this means, you may log in to a file server machine to perform operations not directly available via the basic file transfer mechanisms.

2. Calling the FTP subsystem

A number of options are available when running FTP. The program decides which parts of itself to enable and where user commands will come from by inspecting the command line. The general form of the command line to invoke FTP looks like:

>Ftp/ *global-switches* *host-name* *command-list* CR

All parts of the command line except the subsystem name FTP are optional and may be omitted.

2.1 Global switches

Global switches, explained below, select some global program options such as using the Trident disk instead of the Diablo. The first token after the *global-switches*, if present, is assumed to be a *host-name* (a discussion of which appears later in the description of the Open command). The User FTP will attempt to connect to the FTP Server on that host. After connecting to the server, if a *command-list* is present, an interpreter is started which feeds these commands to the User FTP. When the command list is exhausted, FTP returns to the Alto Executive. If no command list is present, an interactive keyboard command interpreter is started.

Each global switch has a default value which is used if the switch is not explicitly set. To set a switch to *false* precede it with a minus sign (thus FTP/ S means "no Server"), to set a switch to *true* just mention the switch.

Switch	*Default*	*Action*
/S	true	[Server] starts the FTP Server. The Server is not started if the User is enabled and is being controlled from the command line.
/U	true	[User] starts the FTP User. As explained above, the interactive command interpreter or the command line interpreter will be started depending on the contents of the command line.
/C	true	[Chat] starts the Telnet. The Telnet is not started if the User is enabled and is being controlled from the command line, or if the system disk is a Trident.
/T	false	[Trident] sets the system disk to be a Trident drive. The default is 0, but can be changed by following the /T with a unit number between 0 and 7 (thus FTP/T5 means use Trident unit 5). User and Server commands apply to files on this disk but command line input is still taken from Com.cm on the Diablo drive.
/L	false	[Log] causes all output to the User FTP window to also go to the file FTP.log on DP0, overwriting the previous contents. Log is true if the User is enabled and is being controlled from the command line.
/A	false	[AppendLog] enables the log but appends to FTP.log rather than overwriting it.
/E	true	[Error] causes FTP to ask you if you want to continue when a non-fatal error happens during execution of a command line. FTP/ E will cause FTP to recover automatically from non-fatal errors without consulting you.

/R	true	[Ram] allows FTP to use some microcode which speeds things up slightly. If your Alto has no ram, this switch is ignored.
/B	false	[Boot] creates FTP.Boot for distribution to boot servers.
/D	false	[Debug] starts FTP in debug mode.

The rest of the global switches are explained below under "Server Options".

2.2 FTP user log

FTP can keep a log (typescript) file for the FTP User window. The file name is FTP.log. It is always enabled when FTP is being controlled from the command line; otherwise it is controlled by the /L and /A global switches. Some keyboard commands do not treat the user window as a simple teletype, so the typescript for these commands will not be exactly what you saw, alas.

2.3 Using a Trident disk

Starting FTP with the /T global switch causes FTP to access local files on a Trident disk (assuming one is connected) rather than on the standard Diablo disk. Accessing a file on a Trident requires more code and more free storage than accessing a file on the Diablo. Since FTP is very short on space, only a User or a Server FTP is started when the /T switch is set. The default is to start a User FTP, but specifying no user (FTP/T U) or specifying a server (FTP/TS) will start a Server FTP instead.

2.4 Server options

Server options are controlled by global switches and by subcommands of the Server keyboard command. There are currently four options:

Switch	*Default*	*Action*
none		If no server option is specified, retrieve requests (disk to net) are allowed. Store requests (net to disk) are allowed unless the store would overwrite an already existing file. Delete and Rename are not permitted.
/P	false	[Protected] Retrieve requests are allowed. Store, Delete, and Rename are not permitted.
/O	false	[Overwrite] Retrieve requests are allowed. Store requests may overwrite files. Delete and Rename are permitted.
/K	false	[Kill] FTP will return to the Alto Exec when the server connection is closed. A simple form of remote job entry can be performed if the user FTP stores some Executive commands into Rem.cm.

2.5 The FTP display

The top inch or so of the display contains a title line and an error window. The title line displays the release date of that version of FTP, the current date and time, the machine's internetwork address, and the number of free pages on the disk. The error window displays certain error

messages if they arrive from the network (errors are discussed in more detail below). A window is created below the title line for each part of FTP that is enabled during a session (server, user, and telnet).

If the FTP Server is enabled, it opens a window and identifies itself. If a User FTP subsequently connects to this Server, the User's network address will be displayed. The Server will log the commands it carries out on behalf of the remote User in this window. The Server is not enabled when FTP is being controlled from the command line.

The FTP User opens the next window down and identifies itself. The command herald is an asterisk.

The User Telnet opens the bottommost window, identifies itself, and waits for a host name to be entered. The Telnet is not enabled when FTP is being controlled from the command line.

3. Keyboard command syntax

FTP's interactive command interpreter presents a user interface somewhat similar to that of the Alto Executive. The standard editing characters, command recognition features, and help facility (via "?") are available. When FTP is waiting for keyboard input, a blinking cursor will appear at the next character position.

3.1 Directing keyboard input to the User and Telnet windows

The bottom two unmarked keys control which window gets characters from the keyboard. Striking the unmarked key to the right of the right-hand SHIFT key directs keyboard input to the Telnet window. Striking the unmarked key to the right of the RETURN key directs keyboard input to the FTP User window. A blinking cursor will appear in the window that owns the keyboard if the window is awaiting typein.

3.2 Keyboard commands

*Open *host-name* CR

Opens a connection to the FTP Server in the specified host. FTP permits only one user FTP connection at a time. In most cases the word "Open" may be omitted: i.e., a well formed *host-name* is a legal command and implies a request to Open a connection. FTP will try for one minute to connect to the specified host. If you made a mistake typing *host-name* and wish to abort the connection attempt, hit the middle unmarked key (to the right of RETURN).

Ordinarily, *host-name* should be the name of the machine you wish to connect to (e.g., Maxc). Most Altos and Novas have names which have been registered in Name Lookup Servers. So long as a name lookup server is available, FTP is able to obtain the information necessary to translate a known host name to an inter-network address.

If the name of the server machine is not known, you may specify an inter-network address in place of the *host-name*. The general form of an inter-network address is:

network # host # socket

where each of the three fields is an octal number. The *network* number designates the network to which the Server host is connected (which may be different from the one to which the User host is connected); this (along with the "#" that follows it) may be omitted if the Server and User are

known to be connected to the same network. The *host* number designates the Server host's address on that network. The *socket* number designates the actual Server process on that host; ordinarily it should be omitted, since the default is the regular FTP server socket. Hence, to connect to the FTP server running in Alto host number 123 on the directly-connected Ethernet, you should say Open 123# CR (the trailing "#" is required).

*Close CR

Closes the currently open User FTP connection.

*Login *user-name password* CR

Supplies any login parameters required by the remote server before it will permit file transfers. FTP will use the user name and password in the Operating System, if they are there, and logging into FTP will pass them back to the OS in the same manner as the Alto Executive's Login command.

When you issue the Login command, FTP will first display the existing user name known to the OS. If you now type a space, FTP will prompt you for a password, whereas if you want to provide a different user name, you should first type that name (which will obliterate the previous one) followed by a space. The command may be terminated by carriage return after entering the user name to omit entering the password.

The parameters are not checked immediately for legality, but rather are sent to the server for checking when the first file transfer command is issued. If a file transfer command is refused by the server because the name or password is incorrect, FTP will prompt you as if you had issued the Login command and then retry the transfer request. Striking DEL in this context will abort the transfer command.

A user name and password must be supplied when transferring files to and from a Maxc system or an IFS. The Alto FTP Server requires a user-password to be supplied if the server machine's disk is password-protected and if the password in the server machine's OS does not match the password on the disk. Thus if the OS was booted and FTP invoked because a Request-for-Connection was received (which bypasses password checking), FTP will refuse access to files unless a password is supplied. However if the OS was booted normally, FTP assumes that the disk owner (who knew the password) will control access by using the server option switches. The user-name is ignored.

*Connect *directory-name password* CR

Requests the FTP server to *connect* you to the specified directory on the remote system, i.e., to give you owner-like access to it. The password may be omitted by typing RETURN after the directory name. As with Login, these parameters are not verified until the next transfer command is issued. At present, the Connect command is meaningful only when transferring files to or from a Maxc system or an IFS; the Alto FTP server currently ignores connect requests. If the multiple directory feature of the Alto Operating System ever comes into widespread use, this may be changed.

*Directory *directory-name* CR

Causes *directory-name* to be used as the default remote directory in data transfer commands (essentially it causes *directory-name* to be attached to all remote filenames that do not explicitly mention a directory). Specifying a default directory in no way modifies your access privileges, whereas connecting gives you owner access (and usually requires a password). Explicitly mentioning a directory in a file name overrides the default directory, which overrides the connected directory, which overrides the login directory. Punctuation separating *directory-name* from other parts of a remote filename should not be included. For example you might type Directory Alto not Directory <Alto>.

*Retrieve *remote-filename* CR

Initiates transfer of the specified remote file to the local host. The syntax of *remote-filename* must conform to the remote host's file system name conventions. Before transferring a file, FTP will suggest a *local-filename* (generally the same as the *remote-filename* without directory or version), and will tell you whether or not the file already exists on your local disk. At this point you may make one of three choices:

1. Type RETURN to cause the data to be transferred to *local-filename*.
2. Type DEL to indicate that the file is not to be transferred.
3. Type any desired *local-filename* followed by RETURN. The previous *local-filename* will disappear, the new filename will replace it, and FTP will tell you whether a file exists with that name. This filename must conform to Alto conventions. You now have the same three choices.

If *remote-filename* designates multiple files (the remote host permits "*" or some equivalent in file names), each file will be transferred separately and FTP will ask you to make one of the above three choices for each file. At present, only Maxc and IFS support this capability. That is, you may supply "*"s in the *remote-filename* when retrieving files from a Maxc or IFS, but not when retrieving files from another Alto.

*Store *local-filename* CR

Initiates transfer of the specified local file to the remote host. Alto filename conventions apply to the *local-filename*; "*" expansion is not supported. FTP will suggest a *remote-filename*, to which you should respond in a manner similar to that described for Retrieve except that if you supply a different filename, it must conform to the remote file system's conventions. The default *remote-filename* is one with the same name and extension as the local file; the remote server defaults other fields as necessary. If the remote host is a Maxc system or an IFS, then the directory is that most recently supplied in Login, Connect, or Directory commands and the version is the next higher.

*Dump *remote-filename* CR

Bundles together a group of files from the local file system into a *dump-format* file (see the Alto Executive documentation for the dump-file format and more on dump-files in general) and stores the result as *remote-filename*. FTP will ask you for the names of local files to include in the dump-file. Terminate the dump by typing just RETURN when FTP asks for another filename. By convention, files in dump-format have extension .dm.

*Load *remote-filename* CR

Performs the inverse operation of Dump, unbundling a dump-format file from the remote file system and storing the constituent files in the local file system. For each file in the dump-file, FTP will suggest a local file name and tell you whether a file by that name exists on your disk. You should respond in the manner described for Retrieve.

*List *remote-file-designator* CR

Lists all files in the remote file system which correspond to *remote-file-designator*. The *remote-file-designator* must conform to file naming conventions on the remote host, and may designate multiple files if "*" expansion or some equivalent is supported there.

If the *remote-file-designator* is terminated by a comma rather than RETURN, FTP prints a prompt of "**" at the left margin and prepares to accept one or more subcommands. These subcommands request printout of additional information about each file. To terminate subcommand input, type RETURN in response to the subcommand prompt. The subcommands are:

Type	Print file type and byte size.
Length	Print length of file in bytes.
Creation	Print date of creation.
Write	Print date of last write.
Read	Print date of last read.
Times	Print times as well as dates.
Author	Print author (creator) of file.
Verbose	Same as Type Write Read Author.
Everything	Print all information about the file.

This information is only as reliable as the Server that provides it, and not all Servers provide all of these file properties. Altos derive much of this information from hints, so do not be alarmed if it is sometimes wrong.

*Delete *remote-filename* CR

Deletes *remote-filename* from the remote file system. The syntax of the *remote-filename* must conform to the remote host's file system name conventions. After determining that the remote file exists, FTP asks you to confirm your intention to delete it. If the *remote-filename* designates multiple files (the remote host permits "*" or some equivalent in file names), FTP asks you to confirm the deletion of each file.

*Rename *old-filename new-filename* CR

Renames *old-filename* in the remote file system to be *new-filename*. The syntax of the two filenames must conform to the remote host's file system name conventions, and each filename must specify exactly one file.

*Quit CR

Returns control to the Alto Executive, closing all open connections.

*Type *data-type* CR

Forces the data to be interpreted according to the specified *data-type*, which may be Text or Binary. Initially the type is Unspecified, meaning that the source process should, if possible, decide on the appropriate type based on local information.

*Byte-size *decimal-number* CR

Applicable only to files of type Binary, Byte-size specifies the logical byte size of the data to be transferred. The default is 8.

*EOL *convention* CR

Applicable only to files of type Text, this command specifies the End-of-line convention to be used for transferring text files. The values for *convention* are CR, CRLF, and Transparent. The default is CR.

*User ...

Allows you to toggle switches which control operation of the FTP User. There is currently only one, Debug, which controls display of protocol interactions. Warning: this printout (and the corresponding one in the Server command below) sometimes includes passwords.

*Server ...

Allows you to toggle switches that control operation of the FTP Server. The switches are Protected, Overwrite, Kill, and Debug, corresponding to the global switches /P, /O, /K, and /D.

*Telnet ...

Allows you to toggle switches that control operation of the Telnet. There is currently only one, Close, which closes the Telnet connection if one is open, and clears the Telnet window.

4. Command line syntax

The User FTP can also be controlled from the command line. As explained previously, the first token after the subsystem name and *global-switches* must be a legal *host-name*; if the User FTP can't connect to the FTP Server on that host it will abort and return control to the Alto Executive. If a *command-list* follows the host name, the command line interpreter is invoked instead of the interactive keyboard interpreter. This permits the full capabilities of the Alto Executive (filename recognition, "*" expansion, command files, etc.) to be used in constructing commands for FTP.

Each command is of the form:

keyword/switch-list argument ... argument

To get a special character (any one of *#';) past the Alto Executive, you must precede it by a single quote. To get a "/" into an FTP argument, the "/" must be proceeded by two single quotes (the second one tells FTP to treat the "/" as an ordinary character in the argument, and the first one gets the second one past the Alto Executive).

Unambiguous abbreviations of commands are legal. However, when constructing command files, you should always spell commands in full, since the uniqueness of abbreviations in the present version of FTP is not guaranteed in future versions.

A command is distinguished from arguments to the previous command by having a switch on it, so every command must have at least one switch. The switch /C has no special meaning and should be used on commands where no other switches are needed or desired.

4.1 Command line errors

Command line errors fall into three groups: syntax errors, file errors, and connection errors. FTP can recover from some of these, though it leaves the decision about whether to try up to you.

Syntax errors such as unrecognized commands or the wrong number of arguments to a command cause FTP's command interpreter to get out of sync with the command file. FTP can recover from syntax errors by simply ignoring text until it encounters another command (i.e., another token with a switch).

File errors such as trying to retrieve a file that does not exist are relatively harmless. FTP recovers from file errors by skipping the offending file.

Connection errors such as executing a store command when there is no open connection could cause FTP to crash. FTP can't recover from connection errors.

When FTP detects an error, it displays an error message in the User window. If the error is fatal, FTP waits for you to type any character and then aborts, causing the Alto Executive to flush the rest of the command line, including any commands to invoke other subsystems after FTP. If FTP can recover from the error, it asks you to confirm whether you wish to continue. If you confirm, it plunges on, otherwise it aborts. The confirmation request can be bypassed by invoking FTP with the global error switch false (FTP/ E ...) in which case it will plunge on after all non-fatal errors. If you aren't around when an error happens and you have told FTP to get confirmation before continuing after an error, the remote Server will probably time out and close the connection. If you then return and tell FTP to continue, it will get a fatal connection error and abort.

4.2 Command line commands

Open/C *host-name*

See the description in "Keyboard commands". The first token after the subsystem name and global switches is assumed to be a host name and no Open verb is required (in fact, if you supply it, FTP will try to make a connection to the host named Open, which is almost certainly not what you want).

Close/C

Closes the currently open User FTP connection.

Login/C *user-name password*

See description in "Keyboard commands". The *password* may be omitted.

Login/Q *user-name*

Causes FTP to prompt you for the password. This form of Login should be used in command files since including passwords in command files is bad practice.

Connect/C *directory-name password*

See description in "Keyboard commands". The *password* may be omitted.

Connect/Q *directory-name*

Causes FTP to prompt you for the password needed to connect to the specified *directory-name*. This form of Connect should be used in command files since including passwords in command files is bad practice.

Directory/C *directory-name*

See description in "Keyboard commands".

Retrieve/C *remote-filename ... remote-filename*

Retrieves each *remote-filename*, constructing a local filename from the name body of the actual remote filename as received from the Server. FTP will overwrite an existing file unless the /N (No overwrite) switch is appended to the Retrieve command keyword.

If the remote host allows "*" (or some equivalent) in a filename, a single *remote-filename* may result in the retrieval of several files. (Note that one must quote the "*" to get it past the Alto Executive's command scanner.) As noted previously, this capability is implemented only by the Maxc and IFS FTP Servers at present.

Retrieve/S *remote-filename local-filename*

Retrieves *remote-filename* and names it *local-filename* in the local file system. This version of Retrieve must have exactly two arguments. FTP will overwrite an existing file unless the /N (No overwrite) switch is also appended to the Retrieve command keyword. The *remote-filename* should not cause the server to send multiple files.

Retrieve/U *remote-filename ... remote-filename*

Retrieves *remote-filename* if its creation date is later than the creation date of the corresponding local file. A file will be retrieved only if a file with the same name already exists on the local disk. This option may be combined with Retrieve/S to rename the file as it is transferred.

Retrieve/V *remote-filename ... remote-filename*

Requests confirmation from the keyboard before writing a local file. This option is useful in combination with the Update option since creation date is not a fool-proof criterion for updating a file.

Store/C *local-filename ... local-filename*

Stores each *local-filename* on the remote host, constructing a remote filename from the name body of the *local-filename*. A *local-filename* may contain "*", since it will be expanded by the Alto Executive into the actual list of filenames before the FTP subsystem is invoked.

Store/S *local-filename remote-filename*

Stores *local-filename* on the remote host as *remote-filename*. The *remote-filename* must conform to the file name conventions of the remote host. This version of Store must have exactly two arguments.

Dump/C *remote-filename local-filename ... local-filename*

See the description in "Keyboard Commands".

Load/C *remote-filename*

See the description in "Keyboard commands". If the /V switch is appended to the Load command keyword, FTP will request confimation before writing each file. Type RETURN to write the file, DEL to skip it. FTP will overwrite an existing file unless the /N (No overwrite) switch is appended to the Load command keyword.

Delete/C *remote-filename ... remote-filename*

See the description in "Keyboard commands". If the /V switch is appended to the Delete command keyword, FTP will request confirmation before deleting each file. Type RETURN to delete the file, and DEL if you don't want to delete it.

Compare/C *remote-filename ... remote-filename*

Compares the contents of each *remote-filename* with the file by the same name in the local file system. It tells you how long the files are if they are identical or the byte position of the first mismatch if they are not. (No corresponding command is available in the Keyboard command interpreter for implementation reasons: there is not enough room for it in Alto memory.)

Compare/S *remote-filename local-filename*

Compares *remote-filename* with *local-filename*. The *remote-filename* must conform to the file name conventions of the remote host. This version of Compare must have exactly two arguments.

Rename/C *old-filename new-filename*

See the description in "Keyboard commands".

Type/C *data-type*

See the description in "Keyboard commands".

Byte-size/C *decimal-number*

See the description in "Keyboard commands".

EOL/C *convention*

See the description in "Keyboard commands".

Debug/C

See the description of the Debug subcommand of the User command in "Keyboard commands".

4.3 Command line examples

To transfer files FTP.run and FTP.syms from the Alto called Michelson to the Alto called Morley, one might start up FTP on Michelson (to act as an FTP Server), then walk over to Morley and type:

>Ftp Michelson Retrieve/c Ftp.run Ftp.syms CR

Alternatively, one could start an FTP server on Morley (invoking it by FTP/O to permit files to be overwritten on Morley's disk), then issue the following command to Michelson:

>Ftp Morley Store/c Ftp.run Ftp.syms CR

The latter approach is recommended for transferring large groups of files such as *.run (since expansion of the "*" will be performed by the Alto Executive).

To retrieve User.cm from the FTP server running on Alto serial number 123 (name unknown, but it is on the local Ethernet):

>Ftp 123'# Retrieve User.cm CR

Note that the "#" must be preceded by a single quote when included in a command line, since otherwise the Alto Executive does funny things with it. (Quotes are not necessary when typing to FTP's interactive keyboard interpreter).

To start FTP, have the FTP User connect to Ivy, and then accept further commands from the keyboard:

>Ftp Ivy CR

To retrieve <System>Pup-Network.txt from Maxc and store it on the Alto as PupDirectory.bravo, and store PupRTP.bcpl, Pup1b.bcpl, and PupBSPStreams.bcpl on <DRB> with their names unchanged:

>Ftp Maxc Connect/c drb mypassword Retrieve/s <System>Pup-Network.txt
PupDirectory.bravo Store/c PupRTP.bcpl Pup1b.bcpl PupBSPStreams.bcpl CR

To retrieve the latest copy of all .Run files from the <Alto> directory on Maxc, overwriting copies on the local disk (the single quote is necessary to prevent the Alto Executive from expanding the "*"):

>Ftp Maxc Ret/c <alto>'*.Run CR

To update the local disk with new copies of all <Alto> files whose names are contained in file UpdateFiles.cm, requesting confirmation before each retrieval:

>Ftp Maxc Dir/c Alto Ret/u/v @UpdateFiles.cm@ CR

To store all files with extension .Bcpl from the local disk to your login directory on Iris (the Alto Executive will expand *.bcpl before invoking FTP):

>Ftp Iris St/c *.Bcpl CR

To retrieve <System>Host-name/descriptor-file.txt;43 (two single quotes are necessary to get the "/" past the Alto Executive and the FTP command scanner, and one quote is necessary to get the ";" past the Alto Executive):

>Ftp Maxc Ret/c <System>Host-name''/descriptor-file.txt';43 CR

5. File Property Defaulting

Without explicit information from the file system, it is often difficult to determine whether a file is Binary or Text, if Binary, what its byte-size is, and if Text, what End-of-line convention is used. The User and Server FTPs use some simple heuristics to determine the correct manner in which to transfer a file. The heuristics generally do the right thing in the face of incomplete information, and can be overridden by explicit commands from a human user who knows better.

The FTP protocol specifies a standard representation for a file while in transit over a network. If the file is of type Binary, each logical byte is packed right-justified in an integral number of 8-bit bytes. The byte-size is sent as a property along with the file. If the file is of type Text, each character is sent right-justified in an 8-bit byte. An EOL convention may be sent as a file property. The default is that RETURN marks the end of a line.

5.1 File types

FTP determines the type of a local file by reading it and looking for bytes with the high-order bit on. If any byte in the file has a high-order bit on, the file is assumed to be Type Binary, otherwise it is assumed to be Type Text. FTP will generate a warning, but allow you to send what it thinks to be a text file as type Binary, since no information is lost. It will refuse to send a binary file as type text.

Moral: Don't specify a Type unless you know what you are doing. The heuristic will not lose information.

5.2 Byte-size

If a file is type Binary, the byte-size is assumed to be 8 unless otherwise specified. The FTP User and Server will both accept binary files of any byte-size and write them as 8 bit bytes on the disk. No transformation is done on the data as it is written to the disk: it is stored in network default format. Since there is no place to save the byte-size property, it is lost.

Similarly, requests for Binary files will be honored with any byte size, and whatever is on the disk will be sent to the net without transformation. Since Alto files have no byte-size information, the byte-size property will be defaulted to 8 unless otherwise specified, in which case whatever was otherwise specified will be returned as the byte-size.

> *Moral*: Don't specify a Byte-size unless you know what you are doing. Alto-Alto transfers can't go wrong. Alto-Maxc transfers with weird byte-sizes will not work unless the byte-size specified in the Alto to Maxc direction is the same as the byte-size in which the file was stored on the Alto. If it isn't, the Alto will not give any error indication, but the result will be garbage.

5.3 End-of-line conventions

FTPs are expected to be able to convert text files between their own internal EOL convention and the network standard. Conveniently enough, the Alto file system's internal representation of a text file is the same as the network standard. The Alto FTP does not do any transformations on text files. It will refuse to store a text file coming in from the net whose EOL convention is CRLF.

As an escape to bypass conversion and checking, EOL convention Transparent tells both ends *not* to convert to network standard, but rather send a file as-is. This is included for Lisp files which contain internal character pointers that are messed up by removing LFs.

> *Moral*: Don't specify an EOL convention unless you know what you are doing. If your text file is a Lisp source file, specify EOL convention Transparent.

5.4 File dates

The Alto file system keeps three dates with each file: Creation, Read, and Write. FTP treats the read and write dates as properties describing the local copy of a file: when the file was last read and written in the local file system. FTP treats the creation date as a property of the file contents: when the file contents were originally created, not when the local copy was created. Thus when FTP makes a file on the local disk, the creation date is set to the creation date supplied by the remote FTP, the write date is set to "now" and the read date is set to "never read".

6. Abort and error messages

Error and Abort packets are displayed in the system window. Abort packets are fatal; Error packets are not necessarily so.

The most common Abort message is "Timeout. Good bye", generated when a server process has not received any commands for a long time (typically 5 minutes).

The most common Error message is "Port input queue overflowed" indicating a momentary shortage of input buffers at the remote host. Receiving an Error Pup does not imply that the file in transit has been damaged. Loss of or damage to a file will be indicated by an explicit message in the User FTP window. The next iteration of Pup will probably rename Error Pups to be Information Pups.

7. Telnet

FTP provides a simple User Telnet as a convenience for logging into a remote host (e.g., Maxc) to poke around without having to leave the FTP subsystem and start Chat. It lacks most of the creature comforts Chat provides, such as automatic attaching to detached jobs, automatic logging in, etc. The Telnet is not enabled when the User FTP is being controlled from the command line. When the Telnet does not have an open connection, it waits for you to type a host name with the syntax explained above for the Open command, and then attempts to connect to the specified host. If you wish to abort the connection attempt, hit the bottom unmarked key (opposite the right-hand SHIFT key). You can get a larger Telnet window by not starting a server (type FTP/ S to the Executive).