Unicenter® TCPaccess™ FTP Server

User Guide

r6.1 SP2

Computer Associates®

K02215-2E

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

This documentation may not be copied, transferred, reproduced, disclosed or duplicated, in whole or in part, without the prior written consent of CA. This documentation is proprietary information of CA and protected by the copyright laws of the United States and international treaties.

Notwithstanding the foregoing, licensed users may print a reasonable number of copies of this documentation for their own internal use, provided that all CA copyright notices and legends are affixed to each reproduced copy. Only authorized employees, consultants, or agents of the user who are bound by the confidentiality provisions of the license for the software are permitted to have access to such copies.

This right to print copies is limited to the period during which the license for the product remains in full force and effect. Should the license terminate for any reason, it shall be the user's responsibility to return to CA the reproduced copies or to certify to CA that same have been destroyed.

To the extent permitted by applicable law, CA provides this documentation "as is" without warranty of any kind, including without limitation, any implied warranties of merchantability, fitness for a particular purpose or noninfringement. In no event will CA be liable to the end user or any third party for any loss or damage, direct or indirect, from the use of this documentation, including without limitation, lost profits, business interruption, goodwill, or lost data, even if CA is expressly advised of such loss or damage.

The use of any product referenced in this documentation and this documentation is governed by the end user's applicable license agreement.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227-7013(c)(1)(ii) or applicable successor provisions.

© 2005 Computer Associates International, Inc.

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

Contents

Chapter 4: Load Module Transfer (LMTR)

Index

Chapter 1

Introduction to Unicenter TCPaccess® FTP™ Server

This chapter describes Unicenter TCPaccess FTP Server, its capabilities, and the programs it provides, it contains the following topics:

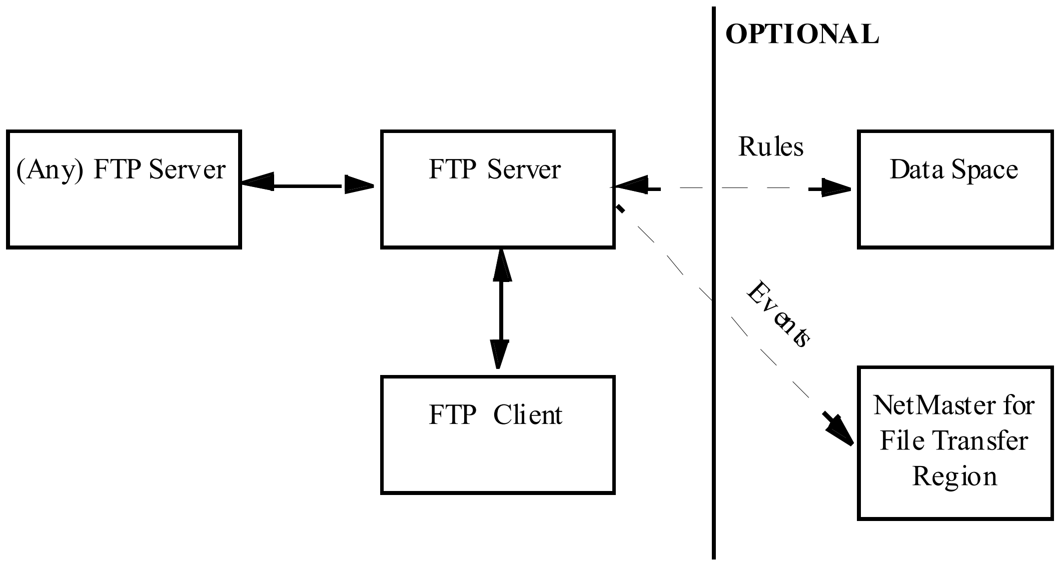
Unicenter TCPaccess FTP Server provides standard FTP services for transferring files over TCP/IP.

Software Components

Unicenter TCPaccess FTP Server consists of the following components:

- An FTP client through which you can request file transfers. The Unicenter TCPaccess FTP Server client can run as a TSO command, as a batch program, or under the OMVS shell.
- An FTP server to perform one side of the data transfer.
- A second FTP server to perform the other side of the data transfer. This can be the same server, although it is normally remote.
- Components running in a Unicenter NetMaster for File Transfer region to monitor the transfer through events sent to it from the server. The server communicating with the region must be in the same operating system image as the Unicenter TCPaccess FTP Server server.
- A data space and associated manager to act as a repository for FTP Policy rules.

Unicenter TCPaccess FTP Server System Architecture



Using Host Name Strings

User FTP programs require you to identify the remote host to which a connection is to be established. Depending on the service to which the connection is established, you might also want to specify optional parameters, such as port number.

Host name strings comprise three sections as illustrated in the following:

```
host<route>,port
```

The only required portion of the host string is the host section.

Specifying the Host

Each host is assigned an Internet host number, or Internet address. Because Internet numbers are hard to remember and keep track of these host numbers, a Domain Name System tracks Internet addresses and correlates them to host names. So you can use names instead of numbers to reference host computers:

`host_name.network_name.`

Example

In the following example, the name identifies a particular host, UNIX, (host number 123.196.222.160) in the COMPANY.NAME.COM system. Notice that the name is terminated with a period. This identifies the name as being fully qualified:

`UNIX.COMPANY.NAME.COM.`

When you enter a fully qualified host string, that is, a host string with the trailing period, Unicenter TCPaccess FTP Server processes it exactly as it was entered. When the name is not fully qualified, Unicenter TCPaccess FTP Server tries to resolve it using a search list set up by the system administrator. If the search list is properly set up, a host string such as HOST1 can be entered and Unicenter TCPaccess FTP Server processes it as if HOST1.COMPANY.COM. had been entered.

Sometimes a host name has an alternate name, or an alias, defined for it. If the target host has an alias, you can enter the alias. HOST1.COMPANY.COM. has the alias UNIX. Entering either of these host strings causes a connection to host 123.196.222.160.

As an option, you can enter the Internet host number instead of the host name. Internet host numbers consist of four integers (between 0 and 255) separated by periods. This is known as dotted decimal notation. Entering 26.0.0.73 specifies the host name NIC.DDN.MIL.

Note: The host parameter of the host string is required.

Specifying the Port

The port option lets you select the port number on the destination host. By default, FTP connects to port 21. If a port variable is used, it must have a leading comma to separate it from the host or <*route*> options.

The port number is an integer with a value between 1 and 65535. If the port number is greater than 999, do not use a comma within the port number. For example, enter 1021, not 1,021.

Note: The port option of the host string is optional.

Example

The following command connects to port 1021 at host hostA:

```
hostA,1021
```

Chapter

2

Client FTP

This chapter describes Client FTP, the File Transfer Protocol (FTP) that allows file transfers among unlike hosts in diverse internetworking environments.

The following topics are included in this chapter:

Introducing Client FTP

Client FTP is a three-party model FTP client. Two control connections are established and maintained by the client. Client FTP connects automatically to the local Unicenter TCPaccess FTP Server and signs on the user.

Connections to the FTP Servers

Client FTP is a true client application. Client FTP uses direct socket connections with both the local Unicenter TCPaccess FTP Server (local server) and the remote FTP server. This eliminates the need for VTAM LU resources, and improves response time.

Throughput and CPU Utilization

Although primarily a three-party client, Client FTP performs some operations in two-party mode to take advantage of the high throughput and low CPU utilization of the Unicenter TCPaccess FTP Server. The result is improved response to the user and quick response time of a direct connection to the remote server.

For file-transfer operations, such as the client commands GET, PUT, and APPEND, Client FTP works in three-party mode. For directory commands, such as the client commands LS and DIR, as well as the implied directory commands in the MGET, MPUT, and MDELETE commands, Client FTP operates in two-party mode.

Client FTP Data Transfer

Client FTP appears to the user as a two-party model by suppressing most local server replies, simplifying the client responses considerably. In addition, Client FTP operates in *blocked* mode. When a data transfer is initiated, the terminal remains blocked until the transfer completes.

Client FTP does not allow terminal input during a data transfer. For long-running transfers, a statistics message is written to the terminal every 10 seconds. You can abort the transfer by pressing the attention key on the terminal keyboard. When the transfer ends, Client FTP displays the reply, and returns to the command prompt.

ISPF Statistics Handling

Unicenter TCPaccess FTP Server provides control over how ISPF statistics are included in the directory entries of qualified target PDS members.

ISPF statistics for the target PDS member can be exported from the source PDS member, or created or updated by the Unicenter TCPaccess FTP Server.

If the end-user wants to export existing ISPF statistics from the source member to the target member, the member **must** be transferred in either compressed or block mode (MODE C or B). Otherwise, the FTP server will create, update or bypass ISPF statistics for target PDS members, depending on the current setting of the ISPFSTATS *state*. This state is initially set to ON or OFF using FTP server configuration. If the FTP server is configured to allow it, end-users can issue a SITE command to toggle the ISPFSATS state for the current FTP session. The format of the SITE command is:

```
SITE ISPFSTATS | NOISPFSTATS
```

ISPFSTATS	Sets the state ON.
NOISPFSTATS	Sets the state OFF.

Note: ISPFSTATS checking applies to data transfers in stream mode (MODE S) only. If the transfer is in compressed or block mode, existing ISPF statistics (if present) are exported from the source PDS member to the target PDS member regardless of the ISPFSTATS setting. If transferring in compressed or block mode and the source PDS member has no ISPF statistics, the target PDS member will not contain ISPF statistics.

Refer to the FTP statement in the *Administrator Guide* for details on the ISPFSTATS configuration parameter and an explanation of server ISPFSTATS behavior in stream mode.

Our FTP client implementation of ISPFSTATS export requires the FTP end-user to issue two additional FTP commands (CD and LCD) to enable our client to obtain the organization of the data sets being FTPed to determine if they qualify.

When exporting existing ISPF statistics from a Unicenter FTP 6.1 client, remote and local change directory commands (CD, LCD) must be issued to alert the Unicenter FTP client to prepare for the transmission of ISPF statistics. Here is an example:

```
mode b
type e
cd 'morwi06.ftp61.samp'
 ---> CWD 'morwi06.ftp61.samp'
250 "'MORWI06.FTP61.SAMP'" partitioned data set is current directory
ftp>
lcd 'morwi06.jcl3'
"'MORWI06.JCL3'" partitioned data set is current directory
ftp>
```

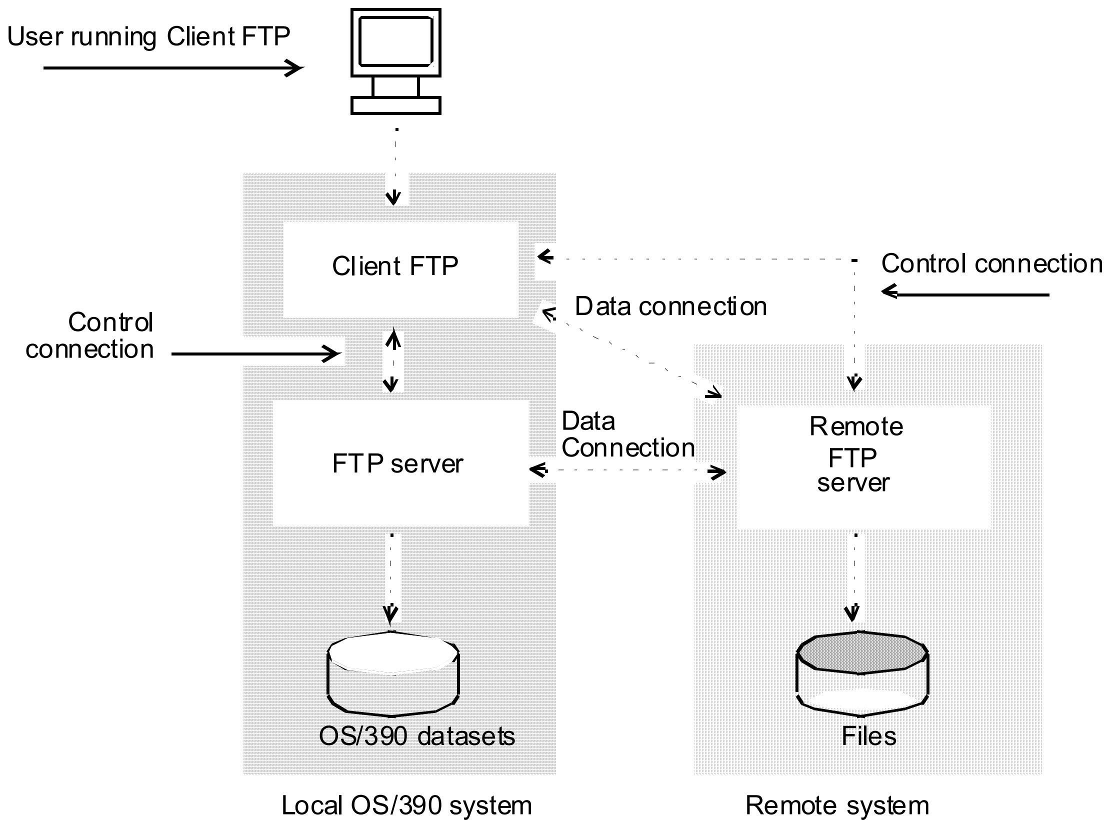
The Unicenter TCPaccess FTP Server client (**not** the FTP end-user) then issues an internal command ("SDIR") to both the local and remote Unicenter TCPaccess FTP Server alerting them that ISPF statistics (if present) must be exchanged. The end user then issues a data transfer command, for example:

```
put mbr1 mbr2
```

Note: In order for ISPF statistics to be exported properly, IBM PTF UW94793 (which fixes APAR OW56783) should be applied to your system.

Client FTP

The following figure shows the relationship between the Client FTP program and the two Server FTP programs in the three-party model.



Note: Two Data Connections are shown. The data connection between the Remote Server FTP and the Local Server FTP is used for the GET, PUT, and APPEND client commands (RETR, STOR, APPE, and STOU server operations). The data connection between the Client FTP and the Remote Server FTP is used for the client LS and DIR commands and the implied LS in the MGET, MPUT, and MDELETE commands (LIST and NLST server operations).

Invoking Client FTP

The Client FTP program runs as a TSO command and can be called as a regular batch program with z/OS and OS/390 JCL.

Invoking Client FTP Through TSO

In a TSO environment, Client FTP can be accessed as a TSO command or it can be called as a program with the TSO CALL command. Because Client FTP does not use full-screen facilities, it can be used from any type of terminal supported by TSO, including 3270 systems, 3767 systems, and asynchronous ASCII terminals supported by NTO or NPSI.

Note: You must have PROMPT set in your TSO profile for Client FTP to work properly in interactive mode.

A left parenthesis "(" separates the remote_host and port_number from the other options.

Example: ftp unix.company.com (translate standard

FTP TSO Command

Some keywords used in the FTP command can be used to override values specified in the TCPIP.DATA data set. See Understanding the Configuration Data Sets for more information.

Invoke Client FTP by entering the FTP TSO command in this format:

```
FTP remote_host [port_number] [options]
```

```
FTP    remote_host
       port_number
       DEBUG
       DOMAIN nn
       EXIT | EXIT=nn
       LOCALHOST host_name
       TIMEOUT nn
       TRACE
       TRANSLATE data_set_name
       VERBOSE
```

Function

Invokes the Client FTP program.

Positional Operands

remote_host	Name of the remote host. Client FTP automatically connects to this host at initialization. This parameter is required. If you do not supply this parameter, then you are prompted for it.
port_number	Port number of the FTP server on the remote host. Default: 21.
DEBUG	Toggle used to activate or deactivate the debugging option. Use the DEBUG or TRACE options interchangeably. You can include either DEBUG or TRACE on the command line, but not both; the second option cancels the first.

Options

DOMAIN	Specifies the domain of the local TCP/IP stack. This defaults to two, and can be specified as another value only when the local stack is Unicenter TCPaccess. This is useful only when running with multiple stacks.
EX*IT* | EXIT=*nn*	Specifies that the client is to terminate in case of an error if the exit_if_error flag is true. Exit=*nn* provides a return code for error conditions.

LOCALHOST *host_name*

Specifies the host name of the local FTP server host.

Note: This option overrides the HOSTNAME statement in the TCPIP.DATA data set.

TIMEOUT *nn*

Sets the following timeout parameters:

MyopenTime

DconnTime

CconnTime

InactTime

DataCtTime

See Understanding the Configuration Data Sets, for the meaning of these timers.

TRACE

Toggle used to activate or deactivate the debugging option. Use the DEBUG or TRACE options interchangeably. You can include either DEBUG or TRACE on the command line, but not both; the second option cancels the first.

TRANSLATE *data_set_name*

Specifies the name of a nonstandard translate table. If this parameter is not supplied, FTP uses the translate table in *hlq*.STANDARD.TCPXLBIN (see TCPIP.DATA for an explanation of the *hlq*).

If present, this parameter is used to construct a data set name in the form *user_id.data_set_name*.TCPXLBIN. If this data set does not exist, FTP attempts to allocate *hlq.data_set_name*.TCPXLBIN.

VERBOSE

Specifies VERBOSE mode. All commands to and replies from the local host are echoed to the user.

TSO CALL Command

Use the TSO CALL command in a TSO environment to call and execute the FTP program out of a specific library. This is especially useful at sites that run multiple releases of the product or have test and production versions of the product at different maintenance levels.

```
CALL       'T01TCP.FTPLOAD(FTP)'
           ['remote_host port options']
```

Positional Operands	T051.V100.LOAD(FTP) Library from which FTP will be called.
remote_host port options	Any number of FTP invocation options can be included in the CALL command. See Invoking Client , for a complete list.
Notes	The data set name, T051.V100, might need to be replaced by the appropriate data set name at your installation. Check with your Unicenter TCPaccess FTP Server site administrator.
	When options are specified in the command statement, they must be enclosed in single quotes.
	When invoked by the CALL command, Client FTP runs as a program and not as a TSO command.
	You can use the NETRC file with the TSO FTP call. See The NETRC File, for more information.

Batch Invocation

The Client FTP program can be run in batch as either a program like any other, or as a TSO command by running it under a batch Terminal Monitor Program (TMP).

You can specify a NETRC file in batch mode. Specify a NETRC DD file with the name of your NETRC file.

Batch Program

You can invoke Client FTP in batch in a manner similar to any other batch utility program. A sample JCL file is contained in the SAMP data set as T051CBJB.

```
//<jobname> JOB job_stmt_parms
//FTPSTEP EXEC PGM=FTP,REGION=1024K,
//            PARM='remote_host port options'
//STEPLIB DD DSN=FTPLOAD,DISP=SHR
//SYSTCPD  DD DISP=SHR,DSN=userid.TCPIP.DATA
//SYSFTPD  DD DISP=SHR,DSN=userid.FTP.DATA
//SYSTERM  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//OUTPUT   DD SYSOUT=*
//SYSIN DD *
unix
user pass
stat
quit
//
```

Optionally, you can include a NETRC file, as shown here:

```
//NETRC DD DISP=SHR,DSN=userid.NETRC
```

Batch TMP

An example of this JCL is located in the SAMP data set as member T051CTMP.

```
//<jobname> JOB job_stmt_parms
//*
//*    RUN FTP UNDER BATCH TSO
//*
//FTP    EXEC PGM=IKJEFT01,REGION=4096K,
//STEPLIB        DD DISP=SHR,DSN=FTPLOAD
//SYSTSPRT       DD SYSOUT=*
//SYSTCPD        DD DISP=SHR,DSN=userid.TCPIP.DATA
//SYSFTPD        DD DISP=SHR,DSN=userid.FTP.DATA
//SYSTERM        DD SYSOUT=*
//SYSTSIN        DD *
FTP remote_host port options
//INPUT    DD *
unix
user pass
stat
quit
//
```

Understanding the Configuration Data Sets

If you have previously installed IBM's TCP/IP for MVS, the configuration files described here might already exist. Unicenter TCPaccess FTP Server provides support for these data sets to allow IBM customers to run their applications.

Client FTP automatically searches for and dynamically allocates the configuration data sets when you start FTP. You can define user-specific environment settings for FTP by setting parameters in these configuration data sets:

- *hlq*.TCPIP.DATA

  TCPIP.DATA defines the Unicenter TCPaccess FTP Server system environment on the local host. You can set your own values by creating a *userid***.**TCPIP.DATA data set.

- *hlq*.FTP.DATA

  FTP.DATA defines the local SITE parameters that are sent to the local host in the form of SITE commands. You can set your own values by creating a *userid***.**FTP.DATA data set.

The high-level qualifier is specified in T051CUM. See Changing the High-Level Qualifier for instructions on changing the default *hlq*.

Samples of these files are located in the SAMP data set. The member names are T051CFTD (FTP.DATA) and T051CTCD (TCPIP.DATA).

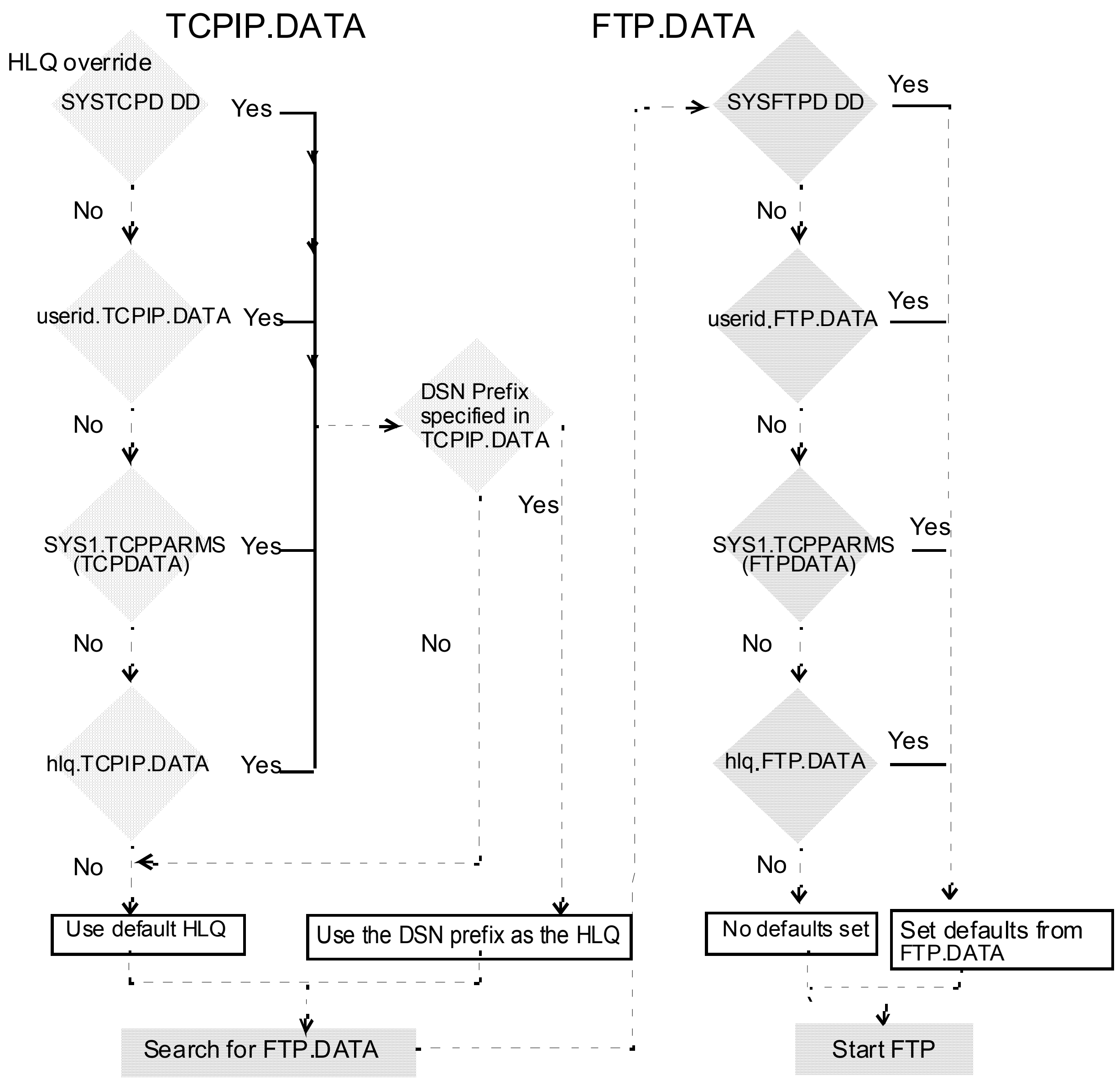
Caution: These configuration data sets must be preconfigured before you execute FTP.

If you want more details about how the configuration data sets are allocated, see the following section, Dynamic Dataset Allocation .

Dynamic Data Set Allocation

Client FTP searches first for the TCPIP.DATA file. If the TCPIP.DATA file is found and contains a DATASETPREFIX statement, then the prefix is used to search for the FTP.DATA file. You can override the search sequence if you want to use high-level qualifiers other than the defaults for the configuration data sets. If the data sets are not found, the search continues as described for each file. Dynamic Data Set Allocation shows the allocation sequence.

Dynamic Data Set Allocation



TCPIP.DATA Allocation

FTP follows the search path described below to locate the TCPIP.DATA file.

1. FTP looks first for a SYSTCPD DD statement, which is used to define an override for the high-level qualifier.
2. If no SYSTCPD DD statement is located, FTP searches for a *userid*.TCPIP.DATA file.

   The user ID is the TSO user ID of the TSO user issuing the FTP command or the user ID of the FTP batch job if the command is issued from a batch job.
3. If no *userid*.TCPIP.DATA file is located, then FTP searches for a TCPDATA member in the SYS1.TCPPARMS partitioned data set (PDS).

   SYS1.TCPPARMS is used during initialization; it may have been created previously if the IBM product was installed.
4. If the TCPDATA member is not located, then FTP looks for the *hlq*.TCPIP.DATA data set.
5. Once the TCPIP.DATA file is located, FTP uses the high-level qualifier defined by the DATASETPREFIX statement in the TCPIP.DATA file.
6. Overriding: If no TCPIP.DATA file is located, then no default values are imposed.

FTP.DATA Allocation

After the TCPIP.DATA search is complete, FTP starts searching for the FTP.DATA file. Once the FTP.DATA file is located, the search is complete.

1. It searches first for a SYSFTPD DD statement.

   The data set name specified in this statement is used by FTP.
2. If no SYSFTPD DD statement is found, then FTP searches for a *userid*.FTP.DATA file.

   The user ID prefix in *userid*.FTP.DATA is the TSO user ID of the TSO user issuing the FTP command or the user ID of the FTP batch job.
3. If no *userid*.FTP.DATA file is located, then FTP searches for an FTPDATA member in the SYS1.TCPPARMS PDS.

   SYS1.TCPPARMS is used during initialization; it may have been created previously if the IBM product was installed.
4. If a *userid*.FTP.DATA is not found, then FTP looks for *hlq*.FTP.DATA.

5. If no FTP.DATA file is located, or if configuration parameters are missing from FTP.DATA, then FTP sets no defaults except for these timeout parameters:

   - CCONNTIME
   - DATACTTIME
   - DCONNTIME
   - INACTTIME
   - MYOPENTIME

   The default for each of these parameters is 300 seconds.

   Normal settings for local SITE parameters are determined by the Unicenter TCPaccess FTP Server administrator in the configuration statements in APPCFG*xx*.

Related References on the FTP.DATA File

For additional detailed information about the FTP.DATA file, see *SC31-7134-01 IBM TCP/IP V3R1 for MVS: Customization and Administration Guide.*

Changing the High-Level Qualifier

Client FTP uses both implicit and explicit data set allocation. You can use the pre-assigned data set names, *hlq*.TCPIP.DATA and *hlq*.FTP.DATA, or you can override the pre-assigned data set names with your JCL.

Client FTP is distributed with the high-level qualifier TCPIP. The name is defined in the T051CUM module. You can change the default high-level qualifier by doing one of the following:

- Apply USERMOD member T051CUM1 in the SAMP data set to change the 26-character field in T051CUM to a value suitable for your site. The APAR contains instructions for changing the high-level qualifier.
- Specify the DATASETPREFIX statement in the TCPIP.DATA data set at execution time.

TCPIP.DATA

The data set TCPIP.DATA defines TCP/IP parameters. An example of this file is located in the SAMP data set member T051CFTD.

FTP does not support multiple host definitions in a single TCPIP.DATA file. If you have a TCPIP.DATA file which contains more than one host definition, you need to modify your file and create a separate TCPIP.DATA file for each host.

DATASETPREFIX *dsn_prefix*	Specifies the high-level qualifier to be used to determine the *hlq*.FTP.DATA data set name and the translate table data set name. Default: None.
HOSTNAME *host_name*	Specifies the host name of the local FTP server. Default: None.
DOMAINORIGIN *origin*	The *origin*, appended to the *host_name*, forms the fully-qualified host name. Default: None.
MESSAGECASE MIXED | UPPER	Specifies whether messages from the FTP client are to be displayed in mixed case or translated to upper case. Default: MIXED.

FTP.DATA

The FTP.DATA data set defines local SITE parameters that are sent to the local host in the form of SITE commands. You can set your own values by creating a *userid*.FTP.DATA data set. An example of this file is contained in the SAMP data set member T051CFTD.

The following parameters are set in the FTP.DATA file. Many of these parameters correspond to SITE commands available in Unicenter TCPaccess FTP Server. You can use the LOCSITE command to change any of these parameters during the FTP session.

AUTOMOUNT TRUE | FALSE	If TRUE, the client sends SITE AUTOMount to local host. **Note:** If FALSE, the client sends SITE NOAUTOmount. Default: None.

AUTORECALL TRUE | FALSE	If TRUE, the client sends SITE AUTORecall to local host; if FALSE, the client sends SITE NOAUTOrecall. Default: None.
BLKSIZE *blk_size* BLOCKSIZE *blk_size*	The client sends a SITE BLKSize=*blk_size* command to the local host. Default: None.
CHKPTINT *checkpoint_interval*	The client sends a SITE CHKPTint=*checkpoint_interva*l command to the local host. Default: 300 seconds**.**
CCONNTIME *close_connection _time*	Set internal timer. Defines the timeout value (in seconds) for closing a control connection. Default: 300 seconds**.**
DATACLASS *SMS_data_class*	Sends a SITE DATAClass=*SMS_data_class* command to the local host. Default: None.
DATACTTIME *data_connection_timeout_time*	Sets internal timer; sends SITE DIDle=*data_connection_timeout_time* to the local host. Defines the timeout value when sending or receiving data. Default: 300 seconds**.**
DCBDSN *DCB_dataset_name*	Sends a SITE DCBdsn=*DCB_dataset_name* command to the local host. Default: None.
DCONNTIME *data_connection_close_time*	Sets internal timer; sends a SITE DClose=*data_connection_close_time* command to the local host. Defines the timeout value when closing a data connection. Default: 300 seconds**.**
DIRECTORY *directory_blocks*	Sends a SITE Directory=*directory_blocks* command to the local host. Default: None.
DIRECTORYMODE TRUE | FALSE	If TRUE, the client sends SITE DIRECTORYmode to the local host. IF FALSE, the client sends SITE DATASETmode. Default: None.

FILETYPE SEQ | JES	Sends a SITE FILEType=SEQ | JES command to the local host. Default: None.
INACTTIME *control_connection_ inactive_time*	Sets internal timer. Defines the timeout for a reply on the control connection. Default: 300 seconds.
LRECL *logical_record_ length*	Sends a SITE LRecl=*logical_record_length* command to the local host. Default: None.
MGMTCLASS *SMS_management_ class*	Sends a SITE MGmtclass=*SMS_management_class* command to the local host. Default: None.
MIGRATEVOL *migrate_volser*	Default: None.
MYOPENTIME *data_connection_ open_time*	Sets internal timer; sends a SITE DOpen=*data_connection_open_time* command to the local host. Defines the timeout when opening a new control or data connection. Default: 300 seconds.
NCP *number_of_channel_ programs*	Sends a SITE NCP=*number_of_channel_programs* command to the local host. Default: None.
PORT local_port	The local host port number used for the control connection. Default: 21.
PRIMARY *primary_space*	Sends a SITE PRImary=*primary_space* command to the local host. Default: None.
RDW TRUE | FALSE	If TRUE, the client sends SITE RDW to the local host; if FALSE, the client sends SITE NORDW. Default: None.
RECFM *record_format*	Sends a SITE RECfm=*record_format* command to the local host. Default: None.

RETPD *retention_period*	Sends a SITE RETpd=*retention_period* command to the local host. Default: None.
SECONDARY *secondary_space*	Sends a SITE SECONDary=*secondary_space* command to the local host. Default: None.
SPACETYPE BLOCKS | CYLINDERS | TRACKS	Sends a SITE BLocks/CYlinders/TRacks command to the local host. Default: None.
STORCLASS *SMS_storage_class*	Sends a SITE STORclass=*SMS_storage_class* command to the local host. Default: None.
UNITNAME *unit_nam*	Sends a SITE UNITNAME=*unit_name* command to the local host. Default: None.
VOLUME *volume_serial*	Sends a SITE VOLUME=*volume_serial* command to the local host. Default: None.
WRAPRECORD TRUE | FALSE	If TRUE, the client sends SITE WRAPRECORD to the local host. If FALSE, the client sends SITE NOWRAPRECORD. Default: None.

General Client FTP Operation

The following steps outline the general procedure for using Client FTP program:

- Issue the FTP command (with any optional parameters) to log on to the remote host. You are automatically logged on to the local host
- If you do not specify a remote host on your FTP command line, then you are prompted for the remote host
- If the remote host name is in the NETRC file, the user ID and password are taken from the NETRC file. If the host name is not in the NETRC file, then you are prompted for a user ID and password

  If your user ID and password fail the logon because one or both were entered incorrectly, you must enter the LOG command to sign on to the remote host
- Set the appropriate file transfer parameters (such as MODE, STRUCT, or TYPE)
- Perform the desired transfer operation (such as GET and PUT)

Path Name

A path name is a string that identifies a file to a file system. A path name must contain a device and/or directory name and a file name. The FTP specification does not specify a standard path name convention. Each user must follow the file naming conventions of the file systems involved in the transfer. For file naming conventions, consult the system administrators at the host sites involved in the transfer.

Many of the Client FTP commands take one or more path name arguments.

For information about the syntax for z/OS and OS/390 path names supported by the Server FTP, see Data Set Names.

Client FTP Command Conventions

The following table provides general notes that apply to the Client FTP commands.

Convention	Description
Program Prompt	To indicate successful completion of most commands, the Client FTP program gives a new prompt.
Completion of Data Transfer	Final completion of the data transfer command is indicated with a message.
Testing the Control Connections	You can use the VERBOSE command to see the specific FTP commands and responses sent and received over the control connections because of Client FTP commands. If you want more information about this test, refer to RFC 959.
Case Sensitivity	The Client FTP commands are not case sensitive.
Abbreviations in Commands	Abbreviations are permitted if they are not ambiguous. For example, you can type AB for ABORT but you cannot type only A because several commands begin with this letter.
Brackets in Commands	In the examples, parameters enclosed by brackets are optional for the command line. In many cases, if the optional parameters are omitted from the command line, you are prompted for them.
Syntax Conventions	Command words are shown in uppercase and parameters are shown in lowercase. When the actual values for a parameter are given (such as LSOUTPUT), they are shown in uppercase.
Example Conventions	In all examples of Client FTP input and output in this manual, user entries are shown in boldface type. The Client FTP examples in this manual assume that you issued OPEN and implied LOG commands similar to this example to connect an IBM z/OS and OS/390 TSO user to another system.

The NETRC File

The Client FTP program uses information in the NETRC file for automatic login to the remote host. The NETRC filename is assumed to be *userid*.NETRC. To use a different naming convention for this file, you must pre-allocate it with an NETRC DD statement.

You can specify any number of MACHINE/LOGIN/PASSWORD sets to define remote hosts. If you define a machine with no login, password, and/or account, you are prompted for this information when needed. When a user connects to a remote host, if a remote MACHINE/LOGIN/PASSWORD set exists for the host, this set logs the user on to the remote host.

If you have not created a NETRC file, you are prompted to supply a user ID and password for the remote host whenever you open a connection to a remote host.

This is the format of the NETRC file; edit this file to specify user IDs, passwords, and accounts, or to create a macro.

```
MACHINE remhost LOGIN userid (for remhost) PASSWORD password (for remhost)
[ACCOUNT account (for remhost)
```

If an installation does not require a password and/or an account, these can be omitted. Before using the Client FTP commands, you must log on to a remote host with an OPEN command.

Caution: You should use your local access control facility to protect NETRC files since these files can contain valid user ID password combinations for remote hosts. Only the TSO user ID using the NETRC file should be able to read their NETRC file.

The NETRC file can also be used in batch jobs by using the NETRC invocation along with a NETRC DD pointing to your NETRC data set.

Note: The batch invocation for NETRC does not assume a default FTP.NETRC data set name.

Client FTP Commands

The following table gives a brief description of each command and its function; detailed descriptions of each command follow the table.

Command	Function
?	Get help on all commands or one command
ACCOUNT	Send account information
APPEND	Append data to file on remote host
ASCII	Set the transfer type to ASCII
BINARY	Set the file transfer mode to binary
CD	Change working directory on the remote
CDUP	Change to parent directory on remote
CLOSE	Disconnect from remote host
CWD	Change working directory (same as CD)
DEBUG	Toggle to activate or deactivate the DEBUG option
DELETE	Delete file on remote host
DELIMIT	Display file delimiter on local host
DIR	List directory on remote host
EBCDIC	Set file transfer type to EBCDIC
GET	Copy file from remote host
HELP	Display help information
LCD	Change working directory on local host
LMKDIR	Create a PDS on local host
LOCSITE	Send SITE to local host
LOCSTAT	Display status for local host
LPWD	Print the local directory name
LS	List file names on remote host
MDELETE	Delete multiple files on remote host
MGET	Retrieve multiple files from remote host
MKDIR	Create a new directory on remote host
MODE	Set transmission mode

Command	**Function**
MPUT	Store multiple files on remote
NOOP	Send NOOP to remote
OPEN	Open connection to remote host
PASS	Send password to remote host
PROMPT	Toggle prompting for MGET, MPUT and MDELETE
PUT	Store file on remote host
PWD	Show name of working directory on remote host
QU*IT*	Exit FTP
QUOTE	Send uninterpreted string to remote host
RENAME	Rename file on remote host
RESTART	Restart a transfer
RMDIR	Remove directory on remote host
RUNIQUE	Toggle STORAGE method on local host
SENDSITE	Toggle sending of SITE command
SITE	Send SITE parameters to remote host
STATUS	Send STATUS to remote
STRUCT	Set file structure
SUNIQUE	Store unique file names on remote host
SYSTEM	Display remote host operating system
TSO	Execute TSO command
TYPE	Set transfer type
USER	Send userid to remote host
VERBOSE	Display system replies with additional information

? Command

Display information about using the Client FTP program. This command is very similar to the HELP command.

```
?[ command_name ]
```

command_name	Command for which you are requesting information.

If the ? command is specified with no *command_name*, then it displays a list of Client FTP commands.

Notes

The ? command can be used either with or without arguments.

If *command_name* is specified as an argument to the ? command, a line of information displays similar to the information in the table in Client . The line shows the command syntax and a short description of the command function.

Example

```
? DIR
DIR pathr pathl - Directory list of remote host
```

! Command

Passes a command to the USS shell for execution. The FTP client must be running under the USS shell for this to be valid.

```
!command
```

command	Specifies the command to pass to the USS shell. There is no space between the ! and the command.

ACCOUNT

Displays information needed by the remote host. See documentation for the remote FTP server for information that it needs.

```
ACCOUNT[ account_info ]
```

account_info	Information to be sent to the remote host.

APPEND

Requests that a file from the local host be appended to a file at the remote host.

```
APPEND[ local_path ] [ remote_path ]
```

local_path	Name of the file to be retrieved from the local host.
remote_path	Remote file to which the local file is to be appended.
	If either file name is omitted, you are prompted for the file name.
Notes	The syntax for each path depends on the associated Server FTP.
	Data set attributes are maintained for the transferred data set.
	If the data set already exists, and the LRECL is less than that of the transferred data set, then the transmitted data set is truncated.

ASCII

Sets the data type to ASCII for the data transfer. This command is equivalent to a TYPE command with the ASCII (A) parameter specified.

ASCIIBINARY

Sets the data type to binary for the data transfer. This command is equivalent to a type command with the image (I) parameter specified.

BINARY	This command has no arguments or keywords.

CD

CD Requests that the remote Server FTP change the current directory to a new directory.

```
CD[ path_name ]
```

path_name

Indicates to the remote Server FTP the name of the directory to be made the current directory.

If you omit *path_name*, the Client FTP program prompts you for the desired value.

Notes

The syntax for *path_name* depends on the associated Server FTP.

If you want to specify a path name that is not a subdirectory of the current directory, enclose the path name in single quotes. Do not leave a space after the first quote, or FTP will ignore the quotation mark.

A UNIX Server FTP in session with the Client FTP program is using /u/user1/work as the current directory. If a CD junk command is issued by the Client FTP to that UNIX Server FTP, the resulting current directory is /u/user1/work/junk. The same result is achieved by specifying CD /u/user1/work/junk.

Example

The following example shows a change of directory to a remote UNIX system:

```
CD /u/lpn/d.ddn
250 CWD command successful.
```

The following example shows a change to a directory on a remote system running Unicenter TCPaccess FTP Server:

```
CD ACCES
250 "'USER1.ACCES.'" is current prefix
```

The following example shows a change to a specific directory:

```
CD ‘TEST.DIR’
250 "'TEST.DIR'" is current prefix
```

CDUP

Directs the remote Server FTP to change the current directory to the parent directory of the old current directory. The CDUP command is most useful when the Server FTP manipulates a hierarchical file system such as UNIX.

Usage

A UNIX Server FTP in session with the Client FTP program has /u/user1/work as the current directory. If a CDUP command is issued by the Client FTP to that UNIX Server FTP, the resulting current directory is the parent of the old current directory (/u/user1).

Example

The following example shows a change to the parent directory of the old current directory on a remote UNIX system:

```
CDUP
250 CWD command successful.
```

CLOSE

Logs you out and terminates the connection between you and the remote Server FTP.

Example

```
CLOSE
221 Goodbye.
```

DEBUG

Displays all commands and responses going to and from the Server FTPs and the user. This command is equivalent to specifying the TRACE option on the Client FTP command line.

Notes

The DEBUG command toggles the previous state. If DEBUG or TRACE was turned on previously, then a subsequent DEBUG or TRACE command turns it off.

DELETE

Directs the remote Server FTP to delete the specified file.

```
DELETE[ path_name ]
```

path_name	The specific file to delete.
	If you omit *path_name*, you are prompted to supply one.
Notes	The syntax for *path_name* depends on the associated Server FTP.

Example

```
DELETE oldfile
250 DELE command successful.
```

DELIMIT

Displays the character delimiter used between the file name and the file type.

The delimiter cannot be changed by this command.

DIR

Requests that the remote Server FTP provide a directory list for the specified path.

```
DIR[ remote_path ] [local_path ] [ ( DISK ]
```

remote_path	Remote directory to be listed.
	If *remote_path* is not specified, then you receive a list of your current directory.
local_path	File to which the directory list is written.
	If *local_path* is not specified, then the directory list is displayed on your screen.
(DISK	Stores output in a data set named *local_directory*.FTP.DIROUTP.
	If *local_directory* is a PDS, then output is stored in member DIROUTP.

Example

The following example lists the contents of a directory on a z/OS and OS/390 system.

```
dir /export/home/user1/mvs
150 ASCII data connection for /bin/ls (138.42.224.15,4126)(0 bytes).
-rw-----       1 user1     dvlp      2730 Oct 22       08:36 channel
-rw-----       1 user1     dvlp      2901 Sep  1       09:17 comten
-rw-----       1 user1     dvlp      1276 Oct 21       14:43 dump
-rw-----       1 user1     dvlp       729 Nov 12       05:24 ibmlink
-rw-----       1 user1     dvlp       751 Nov 12       07:41 ibmlink2
226 ASCII Transfer complete.
```

EBCDIC

Sets the data type to EBCDIC for data transfer. The command is equivalent to the type command with the EBCDIC (E) parameter specified.

Example

```
EBCDIC
EBCD ENTERED
stat
```

GET

Requests that a file from the remote host be copied to a file on the local host by the appropriate Server FTPs. The file to be retrieved is always at the remote host, and the file to be copied into is always at the local host.

GET[*remote_path*] [*local_path*] [(REPLACE]

remote_path	File name to be retrieved from the remote host.
local_path	File name at the local host into which the file from the remote host is copied
REPLACE	Specifies that the data set on the local host file be overwritten.

If you omit either file name, then you are prompted for one.

Notes	The syntax for each path depends on the associated server FTP.

The FTP GET GDGBASE command gets only the most recent data set.

The action of the REPLACE option is dependent on the configuration of the FTP statement in your APPCFG*xx* file:

- If the REPLACE option is specified, it may result in the file on the local host being overwritten. The APPCFG*xx* statements for the local FTP server determine whether the local host file will be overwritten.
- If OVERWRITE is configured, the local host file will be overwritten when the REPLACE option is specified.
- If NOOVERWRITE is configured, the local host file will not be overwritten even if the REPLACE option is specified.
- If SITEOVERWRITE is configured, you must issue a SITE OVERWRITE command and specify REPLACE to overwrite an existing file.

HELP

Requests help information from the local Server FTP.

```
HELP[ ALL | ? | command | [ SERVER [ remote_command ] ] ]
```

HELP	(Without any parameters) requests a list of commands available to the FTP client.
ALL	Provides a brief description of all the commands available to the FTP client.
?	Lists the syntax for the HELP command.
command	Provides a description of a specific command from the local FTP client.
SERVER *remote_command*	Requests information from the remote FTP server.

If a specific *remote_command* is specified, a description of the command is returned from the remote server.

If no command is specified, then a list of supported commands is returned.

LCD

Changes the current working directory on the local host.

```
LCD[ local_path_name ]
```

local_path_name	Specifies the name of the directory on the local host. If you want to specify a path name that is not a subdirectory of the current directory, enclose the path name in single quotes. **Note:** Do not leave a space after the first quote, or FTP will ignore the quotation mark.

LMKDIR

Creates a directory (or PDS) on the local host.

```
LMKDIR[ local_path_name ]
```

local_path_name	Specifies the name of the directory or PDS that is to be created on the local host. If you want to specify a path name that is not to be appended to the current directory, enclose the path name in single quotes. Do not leave a space after the first quote, or FTP will ignore the quotation mark.

LOCSITE

Allows you to specify SITE information to be used by the local host.

```
LOCSITE[ options ]
```

options	Local site information. For a list of the SITE commands, see SITE.

LOCSTAT

Displays local FTP status information.

LPWD

Displays the name of the current working directory on the local host.

LS

Requests a remote Server FTP to provide a list of file names for the specified path.

```
LS[ remote_path ] [ local_path ] [ ( DISK ]
```

remote_path	Path to be listed from the remote host.
local_path	Local file into which the list from the remote server is printed. **Note:** If a local path is not specified, the list of files appears on your screen.
(DISK	*Stores output in a data set named local_directory.FTP.LSOUTPUT. If local_directory is a PDS, output is stored in member LSOUTPUT.*

Notes

If the path specifies a directory or other group of files, the remote Server FTP transfers a list of files.

The syntax for each path depends on the associated FTP server.

If a local path is specified, the list of files is written to the specified file.

Examples

The following example shows of the LS command with parameters:

```
ls /export/home/user1 unix.dir.temp
150 ASCII data connection for /bin/ls (138.42.224.15,4134) (0 bytes).
226 ASCII Transfer complete.
 350 bytes received in 1.54 seconds (227 bytes/s)
```

This is an example of the LS command without parameters:

```
ls
150 ASCII data connection for /bin/ls (138.42.224.15,4135) (0 bytes).
-Transfer complete
channel
comten
dump
filea
tempfile
226 ASCII Transfer complete.
```

MDELETE

Allows you to delete multiple files on the remote host.

```
MDELETE[ remote_path_name ] | [ mask ]
```

remote_path_name Name of the file to be deleted on the remote host.

mask Mask to use to describe files. This mask complies with mask conventions on the remote host.

Example

```
MDELETE userid.myfiles.*
MDELETE file199?
```

MGET

Copies multiple files from the remote host to the local host. You cannot specify a different name for the files that are retrieved. Filenames created on the local host are identical to those on the remote host.

MGET[*remote_path_name*] [(REPLACE]

remote_path_name Name of the files to be copied from the remote host.

REPLACE Specifies that the data set on the local host be overwritten.

Notes The action of the REPLACE option is dependent on the configuration of the FTP statement in your APPCFG*xx* file:

- If OVERWRITE is configured, then the file name is overwritten even without the REPLACE option.
- If NOOVERWRITE is configured, then the file is not overwritten even if the REPLACE option is specified.
- If SITEOVERWRITE is configured, then you must issue a SITE OVERWRITE command and specify REPLACE to overwrite an existing file.

Example

```
ftp> lcd mget.pds
"'GAM2.MGET.PDS'" partitioned data set is current directory
ftp> cd tstdir
---> CWD tstdir
250 CWD command successful.
```

```
ftp> mget TST*
---> PORT 138,42,160,55,17,34
200 PORT command successful.
---> NLST TST*
150 ASCII data connection for /bin/ls (138.42.160.55,4386) (0 bytes).
226 ASCII Transfer complete.
GET TST1
---> PORT 138,42,160,55,17,35
200 PORT command successful.
550 Catalog structure invalid or user lacks authority to catalog
GET TST2
---> PORT 138,42,160,55,17,36
200 PORT command successful.
---> RETR TST2
150 ASCII data connection for TST2 (138.42.160.55,4388) (29 bytes).
226 ASCII Transfer complete.
GET TST3
---> PORT 138,42,160,55,17,37
200 PORT command successful.
---> RETR TST3
150 ASCII data connection for TST3 (138.42.160.55,4389) (37 bytes).
226 ASCII Transfer complete.
ftp>
```

MKDIR

Directs a remote Server FTP to create the specified directory.

`MKDIR[ path_name ]`

path_name

Directory to be created.

If you omit *path_name*, you are prompted for it.

Notes

If the path name is relative, the specified subdirectory is created in the current working directory.

If the path name is absolute, the specified directory is created.

The syntax for path depends on the associated Server FTP.

Example

A UNIX Server FTP in session with the Client FTP program has /u/user1/work as the current directory. If an MKDIR JUNK command is issued by the Client FTP to that UNIX Server FTP, the subdirectory junk is created in the current directory (/u/user1/work/junk). The same result is achieved by specifying **MKDIR /u/user1/work/junk**.

```
MKDIR /u/lpn/d.new
257 MKD command successful.
```

MODE

Sets one of three transmission modes.

```
MODE  S | B | C
```

For the purpose of standardized transfer, the sending host translates its internal end-of-line or end-of-record representation into the representation required by the transfer mode and file structure, and the receiving host performs the inverse translation to its internal representation. Since these transformations make extra work for some systems, identical systems transferring non-record structured text files might use binary representation and stream mode to simplify transfer.

S

Specifies stream mode.

Stream mode passes the data with little or no processing. It interacts with the structure attribute to determine the type of processing.

Stream mode is the default if no MODE command was used.

This is the default if no MODE command was used. In stream mode, the data is transmitted as a stream of bytes. There are no restrictions on the representation type used, and record structures are allowed. In a record structured file, End of Record (EOR) and End of File (EOF) are each indicated by a two-byte control code included with the data sent over the data connection. If the structure is a file structure, the EOF is indicated by the sending host closing the data connection, and all bytes sent over the data connection are data bytes.

B

Specifies block mode.

Block mode formats the data and allows for restart procedures.

In block mode, the file is transmitted as a series of data blocks preceded by one or more header bytes. Record structures are allowed in this mode, and any representation type can be used. Restart markers are embedded in the data stream.

C

Specifies compressed mode. Data transfer type must be set to EBCDIC.

In compressed mode, data blocks are transmitted with one or more header bytes. Logical record boundaries of the data set are preserved with compressed mode. Data is transmitted with no repetitive characters or blanks.

Data compression typically improves network link performance with very little cost in CPU overhead. This feature is most beneficial across networks where link bandwidth may be constrained. Data compression is designed to minimize the amount of data sent across networks by reducing occurrences of repetitive data down to one- or two-byte representations. Printer files, which tend to contain many repeated blank characters, are good candidates for data compression.

The MODE command sets the FTP transmission mode. Only **B** (block) and **S** (stream) were supported in earlier Unicenter TCPaccess FTP Server implementations. Existing FTP MODE command protocol permits you to specify a value of C to signal the FTP client's intent to send or receive compressed data images. If the FTP client does not issue the MODE C command before issuing the actual data transfer command, data compression is not active for the FTP session.

At the end of the FTP session, the FTP client is shown how much data reduction was achieved using compression.

Refer to "Configuring FTP" chapter of the *Administrator Guide* for detailed syntax information.

Notes

One of the three codes (either S, B, or C) is required as an argument on the MODE command.

Each of the possible transmission modes is discussed in the following sections. For a detailed description of the effect of various transmission modes, see the section, *Transmission Modes*, in RFC 959, File Transfer Protocol.

Not all Server FTPs support all transmission modes; review the Server FTP documentation if you have questions concerning transmission mode support.

Variable (RECFM=V) and undefined (RECFM=U) data sets cannot be compressed using a proprietary (non-standard) compression method. These data sets must be compressed using the standard (RFC-compliant) compression method only.

MPUT

Copies multiple files from the local host to the remote host.

You cannot specify a different name for the files that are transferred. Filenames created on the remote host are identical to those on the local host.

```
MPUT[ local_path_name ] | [ mask ]
```

local_path_name	File name on the local host.
mask	Mask to use to describe files. This mask complies with mask conventions on the remote host.

Example

```
MPUT userid.myfiles.*
MPUT file199?
```

NOOP

Tests the response of the remote host.

NOOPOPEN

Opens a connection to the FTP server on the remote host. This command can be issued while you are in the FTP environment.

```
OPENremote_host [ port_number ]
```

remote_host	Name of the remote host. Client FTP automatically connects to this host at initialization. This parameter is required. If you do not supply this parameter, you are prompted for it.
port_number	Port number of the FTP server on the remote host. Default: 21.
Notes	If you are currently connected to a remote host, you must disconnect before using the OPEN command to connect to another host.

PASS

Sends a password to the remote host.

```
PASSpassword
```

password	Specifies your password on the remote host.
Notes	The PASS command must follow the USER command.
See Also	The USER command.

PUT

Requests that the local Server FTP copy a file to the remote system. The file to be copied is always at the local Server and the file destination is always at the remote Server.

```
PUT[ local_pat ] [ remote_path ]
```

local_path	File name of the file to be copied from the local server.
remote_path	File name of the file at the remote server into which the file from the local server is copied. If no name is specified, the name from the local host is used.
Notes	The syntax for each path depends on the associated Server FTP. If you omit either path, you are prompted for the file name.

PWD

Directs a remote Server FTP to return the path name of the current working directory.

PWD This command has no operands.

Example

```
PWD
257 "/u/lpn" is current directory.
```

QUIT

Terminates the Client FTP program.

Example

```
QUIT
221 Goodbye.
```

QUOTE

Sends an uninterpreted, unaltered character string to the remote Server FTP over the control connection. This mechanism sends FTP commands to the Server that the Client FTP program might not be able to send.

`QUOTE[ text ]`

text	Sent to the Server over the control connection exactly as you enter it; if the text is omitted, you are prompted to enter it.

Example

```
QUOTE pasv
227 Entering Passive Mode (26,131,0,17,4,216).
```

RENAME

Directs a remote Server FTP to rename a file.

RENAME[*old_path_name*] [*new_path_name*]

old_path_name	File name to be renamed.
new_path_name	New name to be assigned to that file.

If you omit either argument, you are prompted to enter it.

Notes

The syntax of the path names depends on the associated Server FTP.

Example

```
RENAME titlecol coltitle
350 File exists, ready for destination name
250 RNTO command successful.
```

RESTART

Restarts the last checkpointed file transfer.

Notes

The data set *userid*.FTP.CHKPOINT on the local host holds the valid checkpoint for the last checkpointed file transfer. Any parameters that may have changed for the last transfer, such as transfer mode and type, must be reset before restarting the transfer.

When file transfer is in blocked or compressed mode (mode=b/c), and Client FTP processes a GET or PUT command, it allocates a fixed-block data set named *userid*.FTP.CHKPOINT. It uses this file to save the GET or PUT command and checkpoint record returned from the FTP server. If the *userid*.FTP.CHKPOINT file already exists, then Client FTP rewrites the checkpoint data set from the beginning of the data set.

If the GET or PUT command is successful, the checkpoint data set is deleted upon completion of the command processing. If the GET or PUT command fails or is interrupted, then the *userid*.FTP.CHKPOINT data set is kept in the system for the RESTART command to use to restart the file transfer.

If the RESTART command is executed successfully, then Client FTP deletes the checkpoint data set. If the RESTART command fails or is interrupted, then the checkpoint data set is kept for future use.

Users are responsible for deleting the checkpoint data set once it is determined the data set is no longer needed.

See Also

The RESTART interval option for the SITE command, on SITE.

RMDIR

Directs a remote Server FTP to remove the specified directory.

```
RMDIR[ path_name ]
```

path_name

Directory to be removed.

If you omit *path_name*, you are prompted for it.

Notes

If the path name is relative, the specified subdirectory is removed from the current working directory.

If the path name is absolute, the specified directory is removed.

The syntax of *path_name* depends on the associated Server FTP.

Many systems require the directory to be empty before it can be removed.

Example

As an example, if a UNIX Server FTP in session with the Client FTP program has /u/user1/work as the current directory, and a RMDIR junk command is issued by Client FTP to that UNIX Server FTP, the junk subdirectory of the current directory is removed. The same result is achieved by specifying RMDIR /u/user1/work/junk.

```
RMDIR /u/lpn/d.samp
250 RMD command successful.
```

SENDSITE

Automatically sends the SITE commands when sending a file to a remote host. This commands is useful only with other IBM-based FTP servers.

The SENDSITE command is turned on when you start your FTP session. You can toggle it on and off by issuing the SENDSITE command. To determine the current state of the SENDSITE command, issue the LOCSTAT command.

SITE

(site parameters) Provides the local Server FTP with specific information it requires. This information is essential to file transfers involving that Server FTP, but is not sufficiently universal to have been included specifically in the FTP. Typically, you use a HELP SITE Client FTP command to find the SITE requirements for a specific local Server FTP. Otherwise, review the Server FTP documentation for the SITE requirements.

```
SITE text
```

text	The *text* argument is required and is passed through, unchanged, to the specified server.

STATUS

Requests status information from the remote system.

```
STATUS[ path_name ]
```

path_name	Directory or file name for which information is requested.

STRUCT

(File Structure.) Provides information on file structure to the remote Server FTP.

```
STRUCTF | R
```

F	File structure is specified by F. This is the default if no STRUCT command has been used. File structure is used for files with no internal structure, and the file is considered to be a contiguous sequence of data bytes. File structure is accepted for text files (that is, files with type ASCII or EBCDIC) by all FTP implementations.
R	Record structure is set by R. This is for files made up of sequential records. Default: F if no STRUCT command was used.
Example	`STRUCT F`

SUNIQUE

Stores transferred files by unique file names on a remote machine. SUNIQUE is a toggle command (meaning it is either off or on). When used, the target server automatically ensures that files received in FTP transfer are stored under a unique name.

Notes

The target remote FTP server must support the STOU (store unique) command.

If SUNIQUE is not used, then FTP issues the standard STOR command. If file names are not unique, then files in the target directory could be overwritten.

Default: OFF.

SYSTEM

Displays the name of the operating system on the remote host.

Example

```
SYSTEM
215 MVS is the operating system of this server.
```

TRACE

Activates or deactivates the tracing option. With the tracing option active, you can rerun failing commands to discover the reason for the failure.

Notes

The TRACE command toggles the previous state. If TRACE or DEBUG was turned on previously, then a subsequent TRACE or DEBUG command turns it off.

Example

```
DEBUG
```

TSO

Requests the client FTP program to execute a TSO command for you.

`TSO`*`command`*

command

Specifies the TSO command.

TYPE

Tells a Server FTP the data type to use.

```
TYPEI | L byte_size | { A | E [ N | T | C ] }
```

I

Indicates image type. The data is sent as a contiguous bit stream that, for transfer, is packed into 8-bit transfer bytes. The receiving site stores the data as contiguous bits.

The receiving storage system might need to pad the file (or each record, in record-structured files) to some convenient boundary. Review the documentation for a Server FTP to find out about padding.

Image type is for the efficient storage and retrieval of files and for transfer of binary data. All FTP implementations are required to support the image type.

L *byte_size*

Indicates the local file type and the logical byte size of the file. The byte size value (*byte_size*), representing the logical byte size, is required with the local type. With this type, the data is transferred in logical bytes of the specified size. The logical byte size might differ from the transfer byte size. If the logical and transfer byte sizes differ, the logical bytes are packed contiguously, disregarding transfer byte boundaries, and are padded at the end if necessary.

When the data reaches the receiving host, it is transformed in a manner dependent on the logical byte size and the particular host. The transformation is invertible; an identical file can be retrieved if the same parameters are used.

A

Sets the file type to ASCII. This type is accepted by all FTP implementations and is good for transferring text files, except when both hosts find the EBCDIC type more convenient. In accordance with the *NVT* standard, the CRLF sequence is used at the end of a line of text.

The sender converts the data from an internal character representation to the standard eight-bit NVT ASCII representation (see the TELNET specification in the list of reference documents). The receiver converts the data from this standard form to the receiver's own internal form.

Note: ASCII is the default argument for the TYPE command.

E

Sets the files type to EBCDIC. This type performs efficient transfer between hosts that use EBCDIC. Client FTP users usually use this type when copying files to their z/OS and OS/390 host.

For transmission, data is eight-bit EBCDIC characters. The character code is the only difference between EBCDIC and ASCII types.

End-of-line is rarely used with EBCDIC type to denote structure, but where it is necessary, the NL character is used.

The types ASCII and EBCDIC optionally take a second parameter that indicates what kind of vertical format control, if any, is associated with a file. If a file is to be sent to a host for printing, the receiving host must know how the vertical format control is represented. Therefore, the ASCII and EBCDIC types have a second parameter specifying non-print, TELNET, or carriage control (ASA).

These are the vertical format control specification options:

blank	Move paper up one line.
0	Move paper up two lines.
-	Move paper up three lines.
1	Move paper to top of next page.
+	No movement (that is, overprint).

For both ASCII and EBCDIC file types, vertical format control N is the default.

Notes

One of the four arguments (I, L *byte_size*, A, or E) is required.

If local type (L) is set, then the integer byte size argument must also be set.

If ASCII (A) or EBCDIC (E) type is set, then one of the three vertical format control arguments (N, T, or C) can also be set.

USER

Identifies your user ID to the remote FTP server.

`USER`*`user_id`* `[` *`password`* `] [` *`old_password`* `] [ /` *`new_password`* `]`

user_id Your login name on the remote host.

password Your password on the remote host. You are prompted for your password if you do not supply it.

old_password Your current password on the remote host.

new_password Your new password on the remote host.

Notes

You can issue the USER command to change your user ID at any time while you are in the FTP environment.

Your password is not printed if you allow the client FTP to prompt you for it.

See Also

For more information about using the NETRC file, see The NETRC File.

Chapter 3

Server FTP

This chapter describes the Server program for FTP within Unicenter TCPaccess FTP Server and contains the following topics:

Introducing Server FTP

The Server FTP supports large-scale remote computing on a large IBM (and compatible) mainframe. It includes these types of support:

- File System Support

  The Server FTP supports creation and retrieval of a subset of the disk formats provided by the file system of z/OS and OS/390.

- Record Structure Support

  The z/OS AND OS/390 file system is record-oriented. The FTP Server can translate character files between record structure locally and file structure remotely.

- Binary Files Support

  The Server FTP can transfer large files of binary data efficiently. The parameters required for record-structured binary files (STRU R, MODE B, and TYPE I) are implemented. Restart markers support restarts of large file transfers.

- UNIX System Services (OpenEdition) Support

  Server FTP lets you access UNIX System Services files on machines running Unicenter TCPaccess FTP Server. You may specify z/OS AND OS/390 data sets or UNIX System Services files using the SITE command. If you are not using the SITE command, Server FTP uses the directory to which you are changing to determine the format. If the file specification contains a slash (/), then it is assumed to be a HFS file.

- JES Internal Reader Support

  Server FTP lets data transfer to z/OS AND OS/390 be submitted as a batch job to z/OS AND OS/390 via the JES Internal Reader facility (see the section, JES Internal Reader Support Procedures).

File Handling by the Server FTP

The Server FTP can read an existing disk data set with a wide variety of disk formats and map it correctly into the specified FTP parameters for transmission across the network. The record-structured z/OS AND OS/390 file system forces you to set limits on the size of a record when a file is created. Many processors require this record limit to be a card image (80 characters). A source file prepared on a stream-oriented system and transferred to the mainframe can contain records that are too long, and you may want to specify a larger record size with the SITE command.

Handling a Record That Is Too Long

When the Server FTP receives a file and finds a record too long, it does not discard data; it folds it into multiple records and informs you of its action. As each source language has a different continuation convention, folding the data in this manner is unlikely to match any of these conventions. When the Server FTP preserves data in this manner, you can easily fix the error later.

Individual warning messages are not issued for folded records. The Server FTP counts records folded and sends that count at the end of file transfer (if it is non-zero).

You can also have the Server FTP truncate rather than fold with the ALLO R command. See the ALLO command and the SITE command for details.

Transferring Files to a Host

The FTP lets a character file be transferred to a host for one of three purposes: for printing, for storage and later retrieval, or for processing. Under z/OS AND OS/390, each of these purposes requires a different file format that must be chosen when the file is created.

- By default, the Server FTP assumes processing and records the data in a format that is likely to be acceptable to most z/OS AND OS/390 processing programs (a card image data set).
- When doing a STOR into a data set, the Server FTP infers the purpose from the FTP parameters and disk parameters and performs the appropriate translations.
- To create a print file, a print type must be specified (TYPE AT, TYPE ET, TYPE AC, or TYPE EC).

The translations the Server FTP performs for printing or processing are not exactly *invertible* if the file is later retrieved with FTP. If you want information to be stored for later retrieval in exactly the same form, you must override the default parameters with the SITE command.

Note: An exact representation of the network file is called *raveled*. Raveled files are invertible, meaning you can FTP them back to the originating operating system in exactly the same format they started with. Specify a raveled file when you want to store the file on z/OS AND OS/390 for later retrieval to the originating operating system. In most cases, a raveled file cannot be passed as input to any IBM processing program or sent to the printer.

For a detailed description of raveled and non-invertible files, see Non-invertible Retrieval.

As an aid in setting data set attributes, canned attribute sets for the most common cases can be chosen by mnemonic name on the SITE command. For example, SITE PRINT chooses appropriate attributes for a print file.

Sophisticated File Handling

An experienced user can use additional facilities for more control over Server FTP operations. You can supply explicit Data Control Block (DCB) parameters in the SITE command to cause the FTP Server to create any reasonable sequential data set format used by z/OS AND OS/390 (the Server FTP may also create unreasonable formats).

Also, the Server FTP's default formats for creating new data sets are those generally used by z/OS AND OS/390 programs. You can override these formats to obtain information-conserving storage on the mainframe. The Server FTP lists the full z/OS AND OS/390 data set attributes as well as the FTP transfer parameters when a transfer starts and reports full statistics when the transfer completes.

Transferring Files to a Tape

The FTP Server supports file transfer to and from magnetic tape volumes. This option can be specified dynamically with the FTP Server SITE command described in this chapter.

Configuration

To use this facility, you will need to set the parameters LABEL and MOUNT for the FTP statement in your APPCFG*xx* member of the PARM data set.

To provide installation defaults for tape data set allocation, the system administrator should set up a special GAT TYPE(TAPE) entry in the Generic Attributes Table (GAT) in APPCFG*xx*. Parameters of interest are COMPACT, LABEL(), PARALLELMOUNT, PRIVATE, and UNITCOUNT().

For more information on these parameters, see the *Administrator Guide***.**

SITE Command Parameters

The Server FTP SITE command has parameters specifically for using FTP to transfer files to magnetic tape. These are:

- AUTOINDEX
- COMPACT
- DSEQ
- LABEL()
- MOUNT()
- PARALLELMOUNT
- PRIVATE
- TAPE

These parameters are available for using FTP to transfer files to disk, but may have special significance for transferring files to magnetic tape:

- EXPDT
- RETPD
- UCNT
- VCNT
- VOL(*volser*, *volser*)
- VSEQ

FTP Server Commands APPE and RESTART are not currently supported for the FTP to Tape facility.

For more information on these SITE parameters, see the SITE command.

Using FTP to Write to Magnetic Tape

Cataloging Data Sets

FTP attempts to catalog all data sets created on tape. If the data set name matches an existing name, the transfer will occur but the catalog will not be updated. This may create a problem if a retrieve is issued for the tape version. Therefore, it is recommended that all data sets (disk or tape) have unique names. Use the DELETE command to uncatalog tape data sets.

Cataloged tape data sets are assumed to exist on standard label tapes. Use the SITE command with LABEL and DSEQ parameters when retrieving a cataloged tape data set if the tape does not have standard labels. It is recommended that you use standard label tapes whenever possible.

Writing Multiple Data Sets to Tape

Users who want to write several data sets to tape in one FTP session should be aware that each file to be transferred will generate a mount request, but it may be for a different tape unit. Dynamic allocation does not support RETAIN or UNIT AFFINITY. A workaround would be to arrange with Operations to hard mount a tape and then reference the unit in the SITE command.

Submitting Mount Requests

Server FTP issues the mount request even if running client FTP in batch. This means that Operations should not terminate the mount request by canceling the job requesting the mount. A WTOR message, ACC511A, is issued along with the mount request. A reply of NO to this message cancels the request.

Preventing Timeouts on Data Transfers

FTP will timeout a data transfer request if the remote does not complete the data connection in a certain time. If the remote is another z/OS AND OS/390 system using tapes (or recalling a data set) it will require a tape mount on the remote system before it can complete the data connection. For this reason, FTP will use the longer of MOUNT or HSM times, or 30 minutes if neither tape support nor HSM support is configured.

Using Tape Data Sets on Remote Hosts

The problem of a remote system using tape data sets should also be considered when configuring DATAIDLE time. If a remote is reading a multivolume, for example, it might have to stop the data transfer between volumes while the next tape is mounted. The DATAIDLE time could expire while this is happening.

Server FTP Commands

The following table lists the Server FTP commands. The Unicenter TCPaccess FTP Server program supports most of the FTP commands defined in the FTP specification, RFC 959, *File Transfer Protocol (FTP)*. Server FTP commands are accepted by the local FTP Server when submitted by a remote client. The Unicenter TCPaccess FTP Server replies or responds to the FTP commands listed here.

Note: Not all of the commands listed in this table are documented in this guide. Only those enhanced for Unicenter TCPaccess FTP Server are included. The others in this list conform to their descriptions in RFC 959.

Server FTP Commands Supported by the Unicenter TCPaccess FTP Server

Command	Function
ABOR	Abort
ACCT	Account
ALLO	Allocate
APPE	Append (with create)
CDUP	Change to parent directory
CWD	Change working directory
DELE	Delete
HELP	Help
LIST	List
MKD	Make directory
MODE	Transfer mode
NLST	Name list
NOOP	No operation
PASS	Password
PASV	Passive
PORT	Data port
PWD	Print working directory
QUIT	Logout
REIN	Re-initialize
REST	Restart

Command	Function
RETR	Retrieve
RMD	Remove directory
RNFR	Rename from
RNTO	Rename to
SITE	Site parameters
STAT	Status
STOR	Store
STRU	File structure
TYPE	Representation type
USER	User name

Commands Not Supported by the Unicenter TCPaccess FTP Server

Command	Description
SMNT	Structure mount
STOU	Store unique
SYST	System

The supported FTP commands are described in detail throughout the remainder of this chapter. The HELP command also provides information about the Server FTP. See HELP for guidelines on using the HELP command.

ALLO

The ALLO command allocates a specified amount of disk space for a subsequent STOR or APPE command. When an ALLO command specifies an upper limit on the size of the file as stored in the z/OS AND OS/390 file system, a STOR that starts successfully is guaranteed not to fail because of disk space.

```
ALLO

ALLO

ALLOR logical_record_length

byte_count

byte_count R
logical_record_length
```

Note: The form ALLO R *logical_record_length* is not included in the FTP specification but is a useful extension allowed by the Server FTP.

A comma can replace R.

R	This can be replaced by a comma (,).
logical_record_length	The logical record length to be allocated.
byte-count	The number of bytes to be allocated.

The default, if no ALLO is given, is (5,3) tracks, unless this default has been changed by your Unicenter TCPaccess FTP Server site administrator.

Notes

If an ALLO command is sent, the subsequent STOR or APPE commands operate with these z/OS AND OS/390 SPACE parameters in effect:

T	Floor (byte_count / track_length) (where T is the number of disk tracks needed).
S	max(1, floor(T/5)).

```
SPACE = (TRK,(S+T,S),RLSE) for STOR or SPACE=(TRK,(,S),RLSE) for APPE
```

You can set the space parameters for creating a disk data set either explicitly with a SITE SPACE(..) command or implicitly with an ALLO *integer* command. If both commands are given, the SITE command SPACE parameter takes precedence. If the ALLO *integer* is given after a SITE SPACE(..) command, the reply is: “200 NOTE: Ignored, overridden by site space.”

ALLO R sets the LRECL value for a new data set. Once an ALLO R value has been set, a file received with a record longer than this limit is truncated rather than folded.

An ALLO R value makes sense only with a record-structured file. If the Server receives an ALLO R command when STRU F (file-structure) is specified, the ALLO command fails and returns the reply:

```
“503 Command conflicts with previous commands.”
```

HELP

The HELP command gives you introductory and reference information on the Server FTP. Output from the HELP command is delivered to you by the control connection; the output can be terminated by the Telnet Break facilities. See Telnet Break.

This HELP syntax allows you to request HELP for a *command_name* or *section_title* option, but not with both. When no option is specified, general help information is given:

```
HELP[command_name | section_title]
```

command_name

The command for which help is being requested. Valid command-name strings are ALLO, HELP, REST, SITE, STAT, STRU, and TYPE.

Note: Use the DEFAULT string to request information on the default data set attributes (DCB parameters) created or used by the Server FTP.

section_title

Title of a reference source of introductory or reference information provided through the control connection.

Notes

Each z/OS AND OS/390 site supporting a Server FTP can provide additional help information beyond what is shown here.

Valid *section_title* strings for Unicenter TCPaccess FTP Server are:

```
AECF | AECR | AENF | AENR | AETF | ILF | ILR
```

Use these parameters to request help about Server FTP operation with various TYPE and STRU settings. The following table shows the TYPE and STRU settings corresponding to each parameter:

Using the TYPE and STRU Settings

HELP Parameters	Type	Stru
AECF	AC or EC	STRU F
AECR	AC or EC	STRU R
AENF	AN or EN	STRU F
AENR	AN or EN	STRU R
AETF	AT or ET	STRU F
ILF	I or L	STRU F
ILR	I or L	STRU R

INTRO	Use the INTRO string to request an introduction to the use of the FTP Server.
NEWS	Use the NEWS string to request help on accessing Unicenter TCPaccess FTP Server news.
PATH	Use the PATH string to request information on z/OS AND OS/390 path names (data set names, member names, and volumes) and their relationship to the Server FTP.
SPACE	Use the SPACE string to request information on z/OS AND OS/390 space allocation in relationship to the Server FTP.

MKD

The MKD (MaKe Directory) command creates a partitioned data set (PDS).

```
MKDpathname
```

pathname

Path of the PDS to be created.

The special GAT TYPE(LIBRARY) statement (if present) overrides defaults for the MKD command.

Notes

The pathname can be either a fully or a partially qualified data set name.

These are the possible PDS file attributes:

- Unicenter TCPaccess FTP Server space allocation defaults are SPACE(5,3) DIR(5);
- The DEFGAT initialization statement can provide installation defaults
- Any SITE commands entered override any of the above

Example

```
ftp> pwd
257 "'MVS.'" is current prefix
ftp> mkd mkd.pds
257-"'MVS.MKD.PDS'" partitioned data set created with attributes:
Volser ICSPK1  Unit SYSALLDA  Dsorg PO  Recfm FB  Lrecl 80
Blksize 6160  Space 5 15 Tracks  Rlse  Dir 46
257
ftp> mkd 'mvs.help.pds'
521 "'MVS.HELP.PDS'" data set already exists.
```

Note: See the *Administrator Guide* for information about the GAT statements.

REST

The REST (restart) command specifies that the data transfer command that follows immediately is to restart at a specified intermediate point in the file.

```
RESTrestart_marker
```

restart_marker	Marker from which the restart is to begin.
	Default interval: Every 500,000 data bytes.

Notes

After a REST command, STOR and APPE have identical meanings (APPE is taken to mean STOR).

Data transfer must be in MODE B (block mode). The Server can send and accept restart markers in either STRU F or STRU R.

A file retrieved from the Server FTP includes restart markers at a specific interval. The SITE command RESTART option can change this interval or suppress restart markers entirely. When the count of bytes read from the disk since the last marker reaches the specified interval, a marker is sent at the next end of a complete logical record or segment of a spanned record.

Restart Markers

The following table shows restart markers sent by the Server FTP. These consist of twelve characters that are the ASCII representation of six bytes in the format *VTTRBB.*

Restart Markers Sent by Server FTP

Restart Marker	Description
V	Volume sequence number
TTR	Standard z/OS AND OS/390 disk block address (referred to in IBM publications as a relative track and record address)
BB	A byte offset within a TTR block

RMD

The RMD (ReMove Directory) command deletes an empty PDS. It will not delete a PDS that contains members. To delete a sequential file or a PDS containing members, use DELE.

```
RMD  path_name
```

path_name	Name of the PDS to be deleted.

SITE

The SITE command supplies host-dependent parameters to the Server FTP, for z/OS AND OS/390 data management controls, for special FTP controls, and for generic data set attributes.

```
SITEparameter [ , ]...
```

Available parameters include:

ATTR(*gat_name*)	Specifies any entry in the Generic Attributes table. This command is not supported for the FTP to Tape Facility.
AUTOINDEX	Requests that the data set sequence number be increased by one for each subsequent file transfer.

AUTOMOUNT | NOAUTOMOUNTAUTOMOUNT
Alias for MOUNT; NOAUTOMOUNT is an alias for NOMOUNT.

AUTORECALL | NOAUTORECALL

AUTORECALL	Alias for RECALL.
NOAUTORECALL	Alias for NORECALL.

BLKSIZE (*maximum_physical_block_length*)
BLOCKSIZE (*maximum_physical_block_length*)
LRECL (*logical_record_length*)
or
LINE (*logical_record_length*)
RECFM(*record_format*)

Explicitly sets the DCB or format attributes of a new data set referenced by a STOR or APPE. If the data set is being created, these parameters override the defaults determined by the FTP TYPE or STRU commands. If the data set exists, these parameters must exactly match the corresponding attributes of the data set.

BLOCKSIZE An alias for BLKSIZE. Record formats may be found with the ARECFM parameter.

BLOCK Space allocation is to be in blocks.

CARDS | SOURCE| FORTRAN | OBJECT | LOADLIB | PRINT
Specifies one of the standard Generic Attribute Names supplied with Unicenter TCPaccess FTP Server.

CD | NOCD CD enables directory commands (CWD, PWD, CDUP).

NOCD disables directory commands.

Note: The CD | NOCD parameter is not reset when data transfer begins

CHARSET (*table_name*) Selects an alternate character set (translation) table for single-byte data transfer ASCII data. This table is validated for single-byte data.

CHKPTINT (*checkpoint_interval*)
Specifies the number of logical records between restart markers.

CONDDISP (CATLG | DELETE)
Specifies the conditional disposition for new data sets created by STOR or APPE when the file transfer fails.

COMPACT Specifies IDRC compaction for 3480 tapes.

CYLINDER Space allocation is to be in cylinders.

DACLASS (*sms_data_class*)
Alias for DATACLASS.

DATACLASS (*data_class_name*)
Specifies the SMS data class.

DATASETMODE — Requests the FTP Server to display directory output (LIST/NLST) in data set mode. Each data set is listed individually.

DBCSSET (*table_name*) — Selects an alternate character set (translation) table used for double-byte data transfer ASCII data. This table is validated for double-byte data.

DCBDSN (*data_set_name*) — Specifies a model data set for data set attributes.

DCLOSE (*data_port_close_time*) — Specifies the time, in seconds, FTP will wait to close a data port.

DEVNULL — Requests that the FTP Server allocate a dummy (NULLFILE) data set for storing a data set with the STOR command.

DIDLE (*data_port_idle_time*) — Specifies the time, in seconds, FTP will wait on an idle data port.

Minimum value: 60 seconds.
Maximum value: 1439 minutes.

DIR (*blocks*) — Integer number of 256-byte blocks to be reserved for a PDS directory. One block holds from 7 (load module) to 16 (source module) member entries. This parameter is required to create a new PDS with a STOR or APPE command.

DIRECTORYMODE — Requests that the FTP Server display directory output (LIST/NLST) in directory mode. Data sets, which have the same qualifier at the level immediately below the prefix level, are grouped together as *pseudo-directories*.

The following example shows the use of the DIRECTORY MODE parameter.

```
ftp> quote site directorymode
200 OK, Ready
ftp> dir v*
125 List started OK.
Volume Unit Referred Ext Used Recfm Lrecl BlkSz Dsorg Dsname
ICS009 3390 03/19/96  1    1   VS   29389 29393   PS    VS
Pseudo Directory                                       V191
Pseudo Directory                                       V211
Pseudo Directory                                       V311
Pseudo Directory                                       V410
250 List completed successfully.
```

DOPEN (data_port_open_time) — Specifies the time, in seconds, FTP will wait to open a data port.

DSEQ — Specifies data set sequence number.

DUMMY — This is an alias for DEVNULL.

EXPDT | RETPD

Specifies expiration date and retention period.

EXPDT (*expiration_date*)

Specifies an expiration date for a new data set, in the format:

```
expiration_date = yyyyddd or yyyy/ddd
```

Note: *yyyy* is a year from 1900 to 2155, *ddd* is a Julian date from 1 to 366. You must include any leading zeroes in the *ddd* value.

RETPD (*retention_period*)

Specifies a retention period for a new data set. *retention_period* is a number of days between 1 and 9999.

Note: EXPDT and RETPD are mutually exclusive.

FILETYPE(SEQ | JES | VTOC)

Specifies the type of file the FTP Server is working with:

SEQ	Sequential files. This command is the same as issuing a SITE NOSUBMIT command.
JES	JES spool. Data is written to the JES internal reader. FILETYPE=JES is the same as issuing a SITE SUBMIT command.
VTOC	DASD VTOC records. Directory commands list statistics from the Volume Table of Contents for DASD volumes.

Default: SEQ.

Note: You cannot do a GET of spool files or display the JES spool queue with the DIR command.

FORTRAN | CARDS | SOURCE| OBJECT | LOADLIB | PRINT

See the description for CARDS.

FULLTRK | HALFTRK | VBS | VS

Specifies general attributes for a data set.

HALFTRK | FULLTRK | VBS | VS

See the description for FULLTRK.

HFS | MVS

Specifies the change directory. HFS indicates the change directory should be treated as an HFS filename. MVS indicates the change directory should be treated as an MVS data set.

IBUF (*numbuf bufsize*)

Specifies in sublist notation the number of network input buffers (*numbuf*) and the buffer size (*bufsize*) to be used during data transfer. The maximum value for each number is 32767.

Note: Your Unicenter TCPaccess FTP Server system administrator might have set restrictions on use of the IBUF parameter.

IRBLKSIZE (*max_physical_block*)
IRLRECL (*logical_record_length*)
IRRECFM (*record_format*)

Explicitly set the DCB or format attributes to be used to allocate the internal reader data set when SITE SUBMIT has been entered and a data transfer is performed.

ISPFENQ | NOISPFENQ

Specifies that the ISPF enqueue facility be activated (ISPFENQ) or deactivated (NOISPFENQ).

Default: NOISPFENQ.

ISPFRES | NOISPFRES

Enables (ISPFRES) or disables (NOISPFRES) the RESERVE logic for the SPFEDIT ENQ, if the volume on which the PDS resides is shared by Multiple Systems (UCB shared bit ON). This assures data integrity while the PDS you are accessing is being simultaneously accessed by an ISPF user from another system.

Default: NOISPFRES.

JESLRECL (*record_length*)

Alias for IRLRECL.

JESRECFM (*record_format*)

Alias for IRRECFM.

LABEL

Specifies label type.

Label options supported are: SL, NL, BLP, and LTM, AL. See LABEL parameter description in the description of the GAT statement in the *Administrator Guide*.

LKEDRES | NOLKEDRES

Enables (LKEDRES) or disables (NOLKEDRES) the RESERVE logic for the SYSIEWLP ENQ, if the volume on which the PDS resides is shared by Multiple Systems (UCB shared bit ON). This assures data integrity while the PDS you are accessing is being simultaneously accessed by the linkage editor from another system.

Default: NOLKEDRES.

LINE(*logical_record_length*)

See BLKSIZE.

LISTFMT(OLD | IBM | SHORT)

Specifies whether output from the data set LIST command are in the old FTP format, in IBM-standard format, or in a shortened IBM-compatible format. The short format leaves out data set extents and tracks allocated, but improves LIST response time.

Default: SHORT.

Note: Certain PC-based client FTP packages expect the LIST output from a host configured as OS/MVS to be in standard IBM format.

The LIST parameter is not reset when data transfer begins.

LOADLIB | CARDS | SOURCE| FORTRAN | OBJECT | PRINT

See the description for CARDS.

LRECL (*logical_record_length*)

See BLKSIZE.

MANAGEMENTCLASS (*management_class_name*)

Specifies the SMS management class.

MGMTCLASS (*sms_management_class*)

Alias for MANAGEMENTCLASS.

MIGRATEVOL (*migration_volume_serial*)

The volume serial number of migrated data sets.

MOUNT (*time*)

Enables tape support ability for this file transfer. The time value specifies the maximum wait time in minutes for the tape mount to complete. If the time expires, the request is aborted. MOUNT without a time value specified will use the default system wait time.

NCP (*number_of_DASD_buffers*)

Alias of NDAB.

NDAB(*number1 number2*)

Specifies the number of DASD buffers used by FTP for reading or writing disk data sets.

Maximum value: 99.

Default: Four.

Your Unicenter TCPaccess FTP Server administrator might have set restrictions on the use of the NDAB parameter.

NLSTCASE(UPPER | LOWER)

Specifies whether the output from an NLST command is upper- or lowercase. If LOWER is specified and the data set or member list is part of the current directory, the names are returned in lowercase.

Note: NLSTCASE(LOWER) is supplied to facilitate MGET functions from FTP clients on systems that use lowercase file names. This parameter is not reset when data transfer begins.

OBJECT | LOADLIB | CARDS | SOURCE| FORTRAN | PRINT

See the description for CARDS.

OBUF (*numbuf bufsize*)

Specifies, in sublist notation, the number of network input buffers (*numbuf*) and the buffer size (*bufsize*) to be used during data transfer.

Maximum value for each number: 32767.

Note: Your Unicenter TCPaccess FTP Server system administrator might have set restrictions on use of the OBUF parameter.

OVERWRITE | NOOVERWRITE

This parameter is a toggle. OVERWRITE requests that the FTP Server overwrite an existing data set when transferring files if a data set of the same name already exists on the target Server.

Note: This parameter is:

- Necessary when the SITEOVERWRITE configuration parameter is in effect
- Reset after each transfer, regardless of the PERSIST option in effect

PAD (*pad_code*)

Overrides the default characters that pad network records or lines to fixed-length logical records when data is stored (via STOR or APPE) or deleted from fixed-length logical records when data is retrieved (via RETR). These are the pad-codes:

Z	Pad with zeros.
O	Pad with ones.
B	Pad with blanks.

Defaults: Blanks for character types.
Zeros for binary types.

PARALLELMOUNT

Specifies parallel mounting (mutually exclusive with UCNT).

PDSE | NOPDSE

Allocates PDSEs instead of PDSs, or vice versa.

PERSIST | NOPERSIST Specifies whether SITE parameters are reset following data transfer. If PERSIST is used, all SITE parameters remain in effect until explicitly changed via subsequent SITE commands, or reset with SITE RESET.

Note: If NOPERSIST is used, all SITE parameters are reset after each data transfer.

Default: NOPERSIST.

PRIMARY(*primary_allocation*)

Specifies the primary space allocation for data sets.

PRINT | LOADLIB | CARDS | SOURCE| FORTRAN

See the description for CARDS.

PRIVATE Requests a private volume.

PUSH | POP PUSH causes the Server FTP to save the current settings of parameters entered with previous SITE commands.

POP restores those parameters.

Note: PUSH and POP can be nested. Each PUSH adds an entry to a push-down stack. Each POP pulls the last entry from the stack. When there are no entries in the stack, a POP will result in an error reply.

QDISK (*volume_serial_mask*)

Requests the FTP Server to provide volume information for the volume(s) specified in the *volume_serial_mask*.

Example:

```
ftp> quote site q=ics001
200-              %    Free Free Largest Free
200- Volume Unit Free Cyls Trks Cyls-Trks Exts Address Use Attr
200- ICS001 3390   1   23   102  15   14   23    420   Storage
200 Site command was accepted
```

RDW | NORDW Specifies whether RDWs (Record Descriptor Words) are set as data for RECFM=VB and RECFM=VBS files. If RDW is selected, the RDW is sent for binary, ASCII, or EBCDIC transfers.

Default: NORDW.

Note: RDW will not be translated for ASCII file transfers. This parameter is ignored for data sets with carriage control (RECFM=VBA or RECFM=VBM).

RAVEL | NORAVEL	Specifies whether the file should be raveled.

When the FTP server stores or retrieves a file, the following criteria determine whether the file is raveled:

Note: If the record structure was specified (using the STRU R command), the file is not raveled. Otherwise:

- If SITE RAVEL was specified, the file is raveled
- If SITE NORAVEL is specified, the file is not raveled
- If it is a RETR operation and the input file is a JES spool data set, the file is not raveled
- If the records are blocked (RECFM includes B), the file is not raveled.
- The file is raveled

RECALL(*integer*) | NORECALL

RECALL	Enables Hierarchical Storage Manager (HSM) recall ability for this file transfer. The integer value specifies the maximum wait time in minutes for HSM to complete the recall of the migrated file. If the time expires, the request is aborted. RECALL without an integer value specified enables HSM with the default system wait time.
NORECALL	Disables HSM recall ability for this file transfer.

RECFM(F | FB | FBS | FBA | FBSA | V | VB | VS | VBA | VBS | U))
Record format.

REPLYFMT (OLD | IBM)
Specifies whether reply output, when in FTP/JES mode, is in IBM reply numbers and text as opposed to the Unicenter TCPaccess FTP Server format.

Default: OLD.

RESET — Resets all previous SITE commands. Can be used if SITE PERSIST is specified.

RESTART(*integer*) — When a file is RETRieved in block mode, the FTP Server includes a Restart Marker in the data stream every *integer* data bytes.

Note: To suppress these markers entirely, specify RESTART(0).

For information about restart processing, see the RESTART command and the REST command.

Default: RESTART(500,000).

This command is not supported for the FTP to Tape Facility.

RETPD | EXPDT
See the description for EXPDT.

RLSE | NORLSE

RLSE	Cancels a previous SITE NORLSE. This is needed occasionally to prevent building up too many extents when many APPE operations to the same data set are performed.
NORLSE	Specifies that unused disk space not be released following a STOR or APPE.

Default: RLSE.

SECONDARY (*secondary_allocation*)
Specifies the secondary space allocation for data sets.

SOURCE | CARDS | FORTRAN | OBJECT | LOADLIB | PRINT
See the description for CARDS.

SPACE (*primary_allocation*, *secondary_allocation*)
Specifies primary and secondary disk space allocation (the default allocation is in tracks).

Note: Only required for STOR or APPE commands.

STCLASS (*sms_storage_class_name*)
Alias for STORCLASS.

STORCLASS (*storage_class_name*)
Specifies the SMS storage class.

STRIP | NOSTRIP
Specifies whether pad characters are stripped from fixed-length logical records when data is retrieved (using RETR).

SUBMIT | NOSUBMIT

SUBMIT	Specifies the z/OS and OS/390 FTP Server to send the output of a file transfer to a JES internal reader for execution.
NOSUBMIT	Cancels a previous SITE SUBMIT.

TABS (*integer*)
Specifies the tab stop interval to be used in receiving the next file.

TABS(0) translates tab characters (for example, ASCII x'09' is translated to EBCDIC x'05').

TABS(1) replaces each tab character with a blank.

Default: Eight. Limit is 25.

TAPE	Specifies that attributes taken from the GAT TYPE(TAPE) entry.
TERSE	Requests the FTP Server to issue single-line 150 and 226 replies.
TRACKS	Requests space allocation in tracks.

TRANOPT (*char_translation_mode*)

Defines the character translation mode. These are the choices:

CHAR	Defines character translation mode as single-byte.
DBCS	Defines character translation mode as double-byte.
MIX	Defines character translation mode as single-byte and/or double-byte.

UCNT (*unit_count*)	Specifies the maximum number of generic units an output data set can require. A value of 1 to 59 can be entered.
UNIT (*unit_name*)	Specifies a generic unit for creating a new data set.
UMASK (000)	Allows you to specify a three-character octal number to set file access defaults. **Note:** You must specify this at the beginning of the FTP session.

VBS | VS | FULLTRK | HALFTRK

See the description for FULLTRK.

VCNT (*volume_count*)	Specifies the maximum number of volumes an output data set can require. A value of 1 to 255 can be entered.
VERBOSE	Requests the FTP Server to issue multi-line 150 and 226 replies, showing data set attributes and transfer statistics.

VOLUME (*volume_name*, *volume_name*, ...)

Specifies an explicit volume(s) for creating a new data set or referencing an old uncataloged data set. Normally not required.

A total of 255 can be entered.

Note: If VOLUME is entered with no volume_name, all VOLUME information is reset.

VS | VBS | FULLTRK | HALFTRK

See the description for FULLTRK.

VSEQ (*volume_sequence_number*)

Specifies that processing should begin at a requested volume within a multi-volume data set.

A value of 1 to 255 can be entered.

WRAPRECORD | NOWRAPRECORD

Network records that exceed LRECL is wrapped to the next record when receiving data.

Notes

SITE command keywords are evaluated in left to right order. Successive SITE commands are cumulative. A later SITE command can add to or change attributes established by an earlier SITE command.

SITE command parameters can be entered in either PL/I or BAL formats. The examples for FTP3 are shown in PL/I format. Any SITE parameter that takes a keyword can be entered in either format.

If NOPERSIST is used, all SITE parameters are reset after each data transfer. If PERSIST is used, all SITE parameters remain in effect until explicitly changed via subsequent SITE commands, or reset with SITE RESET.

When a sub-parameter takes a list and more than one value is contained in the list, the list of values must be enclosed in parenthesis. Even if you choose to enter the command in BAL format, you must use parentheses around the list.

The SITE command verb is followed by a list of keyword parameters. Each keyword may normally be abbreviated to the minimum-length unambiguous string, as in TSO. (In the preceding list, the minimum abbreviation for each parameter is shown in uppercase.)

SITE parameters must be separated by a comma or a blank.

A single FTP command is limited to 80 characters. In the (unlikely) event that a SITE command exceeds 80 characters; it can be broken into two or more successive SITE commands.

If an error is found in parsing a SITE parameter, an error message is issued indicating the bad parameter, and the FTP Server continues with the next parameter.

HFS, MVS, and UMASK are persistent for the session. If you do not specify HFS or MVS on the SITE command, then Server FTP uses the change directory to determine the format.

Examples

The following example illustrates the BAL format:

```
SITE ATTR=gat_name
```

The following example illustrates the PL/I format:

```
SITE ATtr (gat_name)
```

The following example illustrates the obligatory use of parentheses when the command is entered in BAL format:

```
SITE VOLUME=(vol1, vol2, vol3)
```

STAT

Provides partial or complete status of the Server FTP.

```
STAT  selector
      item_numbers
      path_name<@font:@>
```

selector A string containing any subset of the letters F | A | P | T | I

F	FTP parameters, such as MODE and TYPE.
A	Access control, such as user ID and account.
P	Path data, such as data set name and attributes.
T	Data transfer status (such as number of bytes, records transmitted); null if no transfer is in progress
I	Internal control blocks of Server FTP.

item_numbers One or more positive integers separated by commas.

path_name A valid z/OS and OS/390 directory identifier, optionally enclosed in quotes.

Notes The commands STAT or STAT ? mean STAT ? FAPT.

The command STAT ? *item_numbers* gives only the specified item of the full STAT display. This is useful in debugging since the full display can be lengthy.

The command STAT *path_name* (where *path_name* is a valid z/OS and OS/390 data set prefix specified in the form *myuid.*) gives catalog information on a specific group of data sets. STAT *path_name* gives the same information as LIST but sends it over the Telnet control connection instead of a data transfer connection.

The command STAT * gives the catalog list for the default (logged-in) directory.

Examples

```
STAT ?
STAT ? selector
STAT ? item_numbers
STAT path_name
```

Data Set Attributes

This section describes the attributes of data sets that can be retrieved (read) or stored (written) by the Server FTP.

Units and Volumes

The Server FTP can write and retrieve only disk data sets stored on permanent-resident disk volumes.

Data Set Names

The Server FTP rules for naming and accessing data sets described here are the same as those for TSO.

User disk data sets generally have names of the form:

```
defprfx.name1[.name2[.name3 .. [.namen]]]
```

defprfx	That portion of the data set name that is the defined default prefix of the installation.
namex	Data set name indexes, made up of one- to eight-alphanumeric characters, the first of which is alphabetic.

Note: The total length of the data set name, including the periods between indexes, cannot exceed 44 characters.

Data set naming conventions can vary between z/OS and OS/390 sites. Consult personnel at your site to learn naming conventions.

When the data set name is enclosed in single quotes, it is a fully qualified data set name. When the data set name is not enclosed in quotes, it is partially qualified. Under this Server FTP (and TSO), you normally specify data set names in a partially qualified fashion, allowing the system to prefix the installation's default prefix to the data set name.

Server FTP uses one of the three possible default prefixes:

- User ID
- None
- Character strings

If User ID is defined at your installation, then Server FTP uses the User ID provided by the USER FTP command as the *defprfx* for prefixing. If no prefixing is defined, the data set is fully qualified and quotes are not required. If character string is defined, the installation selected a common qualifier for all data sets. Individuals can select their own data set prefix with the Change Working Directory (CWD) or Change to Parent Directory (CDUP) FTP commands.

Example

These are some examples of how to set your prefix. The first method shows the cd and cdup User commands; the second shows sending the cwd and cdup commands to the Server using quote.

```
ftp > pwd
257 "'MYID.'" is current prefix.
ftp > cd level1
250 "'MYID.LEVEL1.'" is current prefix.
ftp > cdup
200 "'MYID.'" is current prefix
ftp > cd level2
250 "'MYID.LEVEL2.'" is current prefix.
ftp > cd 'newid'
250 "'NEWID.'" is current prefix.
ftp > quote cwd 'nextid'
250 "'NEXTID.'" is current prefix.
ftp > quote cdup
200 No prefix defined.
```

The Server FTP supports both simple sequential and partitioned data sets. A PDS contains an internal directory to a set of sub-data sets called members. All members of a PDS share the same data set name (*dsname*) and attributes.

The fully qualified name of a PDS member is:

```
defprfx.name1[.name2[.name3 .. [.namen]]](member_name)
```

The *member_name* field has the same syntax as *name1* through *namen*.

The Server FTP also supports Generation Data Group (GDG) data sets. A GDG data set has a similar format to a PDS but the member name takes the form 0, **+***n*, or **-***n*, where *n* is the relative generation number.

FTP Path Name Syntax

The data transfer commands STOR, APPE, RETR, DELE, RNFR, and RNTO have *path_name* as a parameter.

With this Server FTP, *path_name* specifies a data set or PDS member in one of these forms:

name1[.name2[.name3 ... [.namen]]]

name1[.name2[.name3 ... [.namen]]](member_name)

name1[.name2[.name3 ... [.namen]]](GDG_number)

'*name1[.name2[.name3 ... [.namen]]]*'

'*name1[.name2[.name3 ... [.namen]]](member_name)*'

'*name1[.name2[.name3 ... [.namen]]](GDG_number)*'

The first three forms are partially qualified data set names. An installation defined default prefix is prefixed to names in these forms. Typically, you use one of these forms to refer to your own data sets. The last three forms are fully qualified and are used as is by the FTP Server. The second and fourth forms are used for PDS libraries and the third and sixth forms are used for GDGs.

In addition, a *member_name* can be entered alone when the current directory is a partitioned data set. See VTOC and Catalog.

Note: The length of the one to eight-character member name or the GDG number and the enclosing parentheses are not included in the 44-character limit.

Using Wildcard Characters in FTP

The Server commands STAT, LIST, and NLST accept a data set or member name mask as a parameter. The output from the command lists all data sets or members that match the mask criteria.

These are the rules for masking:

- An asterisk (*) represents zero or more consecutive characters
- A percent sign (%) represents a single character

Note: If a member name or member name mask is included, the data set name must not include a mask.

Example

```
 ftp> ls v*.obj
200 OK, Ready
125 Transfer started
V111.OBJ
V20.OBJ
V201.OBJ
226 Transfer complete
ftp> ls v2%.obj
200 OK, Ready
125 Transfer started
V20.OBJ
226 Transfer complete
ftp> cd v20.obj
250 "'MVS.V20.OBJ'" partitioned data set is current directory
ftp> ls ftps*
200 OK, Ready
125 Transfer started
FTPS
FTPSFTDR
226 Transfer complete
ftp> ls v*.obj(*)
200 OK, Ready
501 Wildcard characters are not permitted within a partitioned data set name
```

Partitioned Data Sets

When a CWD or CDUP command causes the prefix to match the data set name of a cataloged PDS, the PDS becomes the working directory. Subsequent data transfer commands STOR, RETR, DELE, RNFR, and RNTO, as well as the LIST, NLST, and STAT commands, will treat unquoted path names as PDS member names. In addition, the LIST and NLST commands without a path name cause a list of members for the PDS directory to be output.

Examples

```
 ftp> pwd

257 "'MVS.'" is the current prefix
ftp> cd help.pds
250 "'MVS.HELP.PDS'" partitioned data set is current directory
ftp> dir
200 OK, Ready
125 Transfer started
  Name     VV.MM  Created    Changed     Size  Init  Mod  Id
FTPDEFAU   04.00  12/08/93   12/09/93    6:16   98    98  0 MVS
FTPINTRO   01.00  12/09/93   12/09/93    6:18  134   134  0 MVS
FTPNEWS    01:00  12/09/93   12/09/93   10:30   25    25  0 MVS
GREETING   01:00  12/08/93   12/09/93    6:19   12    12  0 MVS
226 Transfer complete
ftp> get ftpnews
200 OK, Ready
150-Data set open with attributes:
Type A N Tabs 8 Stru F Mode S Path MVS.HELP.PDS(FTPNEWS)
Volser MVSVOL Unit SYSALLDA Dsorg PO Recfm FB Lrecl 80
Blksize 3120 Rlse
150
226 Transfer complete
```

In addition, the LIST, NLST, and STAT commands accept member name masks as path names. The member name mask can be entered either alone (if the current directory is a PDS) or as part of a fully qualified PDS data set name.

```
ftp> ls 'mvs.help.pds(ftp*)'
200 OK, Ready
125 Transfer started
'MVS.HELP.PDS(FTPDEFAU)'
'MVS.HELP.PDS(FTPINTRO)'
'MVS.HELP.PDS(FTPNEWS)'
226 Transfer complete
```

If you want to treat a PDS data set path name as a prefix, then enclose the fully qualified name in quotes and append a period (**.**) at the end.

```
ftp> cd 'mvs.help.pds.'
250 "'MVS.HELP.PDS.'" is current prefix
```

VTOC and Catalog

Under z/OS and OS/390, each disk volume contains a file directory of the data sets on that volume called the volume table of contents (VTOC). Hence, any disk data set can be located via the logical path name: *volume,dsname*. The Server FTP lets you specify the volume name in the SITE command. See the SITE command on SITE for details.

z/OS and OS/390 have a central file directory, called the catalog that provides the volume name as a function of the *dsname*. TSO generally requires that all user data sets except scratch files be cataloged (that is, listed in a system catalog). A cataloged data set name is unique on the system while an uncataloged data set name need be unique only on the disk volume where it resides. You need to give only the *dsname* (and *member_name* for a PDS), not the volume, to locate an existing cataloged data set.

This Server FTP generally does not change the catalog status of a data set operated on by a STOR, APPE, RETR, or rename operation. A data set created by an FTP STOR, APPE, or RNTO operation is cataloged. If the *dsname* conflicts with an existing cataloged data set, the operation is refused in the case of RNTO.

When the Server FTP performs an APPE or RETR operation on an existing uncataloged data set (using SITE VOLUME), it tries to catalog it. If the attempt fails because another data set with the same name is being cataloged or because the *dsname* has the wrong tree structure, a warning message is sent but the operation proceeds.

z/OS and OS/390 Space Allocation

z/OS and OS/390 allocate disk space to a data set in variable-sized pieces called extents. Each extent is a contiguous set of disk tracks. The byte capacity of a disk track depends on the disk model. For example, A 3380 holds a maximum of 47476 bytes per track. A data set used by the Server FTP is limited to a maximum of 16 extents. Therefore, if the disk space is fragmented, you might run out of extents before running out of total space. In such a situation, choice of appropriate space parameters is important.

Space allocation may have two parameters to the operating system: a primary space quantity, and a secondary space quantity. When an FTP STOR operation is requested, FTP first tries to allocate the primary quantity. If it succeeds, data transfer starts. Each time that space is exhausted during the STOR (or APPE) operation, FTP tries to allocate the secondary quantity. The STOR (or APPE) fails at that point if it cannot satisfy the request. Each allocation, whether primary or secondary, takes place as follows:

- z/OS and OS/390 try to find a contiguous area (extent) to satisfy the request using the best-fit algorithm
- If no single area is large enough, it uses the fewest extents possible (up to five) to satisfy the request

This allocation process continues until the total space or the extent limit is exhausted.

The default space parameters for FTP STOR are (primary, secondary) = (5,3) tracks. The maximum space that can be allocated with these parameters depends on the degree of fragmentation of the volume (assuming that enough total space exists). It ranges from 50 tracks down to 14 tracks (if the largest contiguous area is only 1 track and extents are exhausted first).

These methods obtain more space or a larger file:

- Send an ALLO command before the STOR.

  The parameter to ALLO is a file size in (network) bytes. The Server FTP converts that value to disk tracks and uses 1.2 times that value as the initial space allocation. Therefore, if the data transfer starts, you know the initial space allocation was successful and you cannot run out of space specified by the ALLO. The 1.2 factor is intended to take care of inter-record gaps. The secondary space quantity is 0.2 times the ALLO value.

- Specify explicit SPACE parameter on the SITE command.

  The disk space allocation (ALLO) is in terms of the data stored on disk, including padding. A text file normally is padded to 80 byte records, so each line is 80 bytes on disk. The primary and secondary space quantities are recorded in the data set label (DSCB) by the operating system. A subsequent APPE uses the secondary quantity determined when the data set was created unless you override it with a new ALLO or SITE SPACE before the APPE.

Multivolume Data Sets

If a data set might require more space than is available on a single volume, you can specify that it may reside on multiple volumes using the **S**ITE VOLUME, VCNT, or UCNT parameters. Enough space must exist on the first volume for the primary extent. When the space on the first volume is exhausted, extents are allocated on the next volume. Up to 16 extents can be allocated on each volume.

FTP will use the greatest of the following numbers to determine how many devices and volumes to allocate to a data set:

- Unit-count specified in the SITE UCNT parameter
- Volume-count specified in the SITE VCNT parameter
- Number of serial numbers specified in the SITE VOLUME parameter

Data Set Organization

FTP supports sequential (DSORG=PS) and partitioned (DSORG=PO) data sets. Direct access (DSORG=DA) data sets can be read sequentially but cannot be written by the Server FTP. ISAM and VSAM data sets are not supported.

Disk Format (DCB) Attributes

A data set under z/OS and OS/390 has the DCB attributes listed in the following table.

The first column shows the SITE command keyword parameter(s) to set the corresponding DCB attribute:

Keyword	Corresponding data set (DCB) attribute
RECFM	Record Format (RECFM)
LRECL or LINE	Logical Record Length (LRECL)
BLKSIZE or BLOCK	Physical Block Length (BLKSIZE)

When an existing data set is retrieved, the Server FTP determines the attributes from z/OS and OS/390 and uses them to read the disk data set correctly. However, when writing to a data set, through a STOR or APPE operation, you may need to be concerned about setting the correct attributes. Whether Server FTP assigns attributes for an existing data set depends on whether the data set is partitioned (PDS) or sequential.

Sequential Data Sets

A STOR or an APPE to a non-existent data set creates a new data set. The Server FTP assigns the attributes of the new data set. Additionally, the STOR operation assigns new attributes to an existing data set.

New attributes are set from default attributes chosen by the Server FTP based on the TYPE and STRU transfer parameters (see Generic Attribute Names), or from explicit overrides of these defaults by SITE parameters.

Partitioned Data Sets

A STOR of a member into an existing PDS adds information to the data set (similar to an APPE). The attributes of the data set do not change. Any attributes set by the SITE command must match those of the existing PDS, or the Server FTP responds with this message:

```
554 SITE LRECL, BLKSIZE or RECFM do not match those of existing data set
```

An APPE of a new PDS member is identical to a STOR. However, you cannot APPE into an existing member because the file system replaces the member. If you attempt to APPE into an existing member name, Server FTP responds with this message:

```
504 Not implemented for that parameter, ignored.
```

If a new PDS is being created, by STOR or APPE, new data set attributes are assigned by the rules described in Sequential Data Sets.

Default Data Set Attributes

The Server FTP chooses default data set attributes for STRU based on the TYPE and chooses STRU transfer parameters as shown in the following table:

Type	STRU F	STRU R
AN or EN	SOURCE	SOURCE
AT or ET	PRINT	PRINT
AC or EC	PRINT	PRINT
I or L	VS	VBS

Note: In the STRU R case, the default data set attributes depend on the size passed by the ALLO command. If the ALLO command has not been specified or if ALLO R is less than 81, the default data set attribute is SOURCE. Otherwise (ALLO R is greater than 80), the default data set attribute is VBS, with the LRECL set to ALLO R plus four.

Generic Attribute Names

You can explicitly provide a subset of DCB attributes and DD statement fields on a SITE command. Alternatively, the SITE command can specify one of the generic attribute names given in the following table.

Attribute Name	RECFM	LRECL	BLKSIZE
FULLTRK	FB	80	full-track (see the first note following this table)
HALFTRK	FB	80	half-track (see the first note following this table)
LOADLIB	U	0	*rcmd* (see the second note following this table)
OBJECT	FB	80	*rcmd*
PRINT	VBA	133	*rcmd*
SOURCE CARDS FORTRAN	FB	80 (default)	*rcmd*
VBS	VS	*rcmd*-4	*rcmd*
VS	VS	*rcmd*-4	*rcmd*

Note: Actual values for half- and full-track blocking depend on the output disk device type. These generic types create PDS libraries suitable for object modules and load modules.

Each generic attribute name includes values for RECFM, LRECL, and BLKSIZE, and optionally, UNIT, VOLUME, or SPACE, but the attributes set by a generic attribute name can be overridden by other generic or specific attribute keywords. When overriding SITE command keywords, the keywords are interpreted in left-to-right order.

In addition, other generic attribute names can be defined by your site. These names can stand for other combinations of RECFM, LRECL, BLKSIZE, UNIT, VOLUME, and SPACE. User-defined names must specify RECFM, LRECL, and BLKSIZE. They can optionally specify UNIT, VOLUME, or SPACE parameters.

The generic types SOURCE, CARDS, FORTRAN, OBJECT, LOADLIB, and PRINT can also carry UNIT, VOLUME, and SPACE parameters as defined by your site. They can be referenced by the SITE keywords SOURCE, CARDS, FORTRAN, OBJECT, LOADLIB, and PRINT. Other generic types may have been defined by your site. They can be referenced by the SITE keyword ATTR (type).

Note: In the preceding table, the block size is sometimes specified as *rcmd*, meaning a recommended value. This means the Server FTP chooses an actual default BLKSIZE that is optimum for the particular device on which the data set is allocated and less than or equal to a recommended size. The recommended size is a site-specific FTP parameter. If your site has not changed this parameter, the recommended size is 6K (6144) bytes. The choice of optimum BLKSIZE <=rcmd depends on RECFM, LRECL, and the disk device type.

Rules for Record Formats

These rules govern Server FTP support of the record format (RECFM) data set (DCB) attribute:

- The Server FTP writes into a disk data set using any of the record formats (RECFM) listed in the following table.

Record format	Description
U, F, V, VS	Unblocked
FB, FBS, VB, VBS	Blocked, nonprint format
FBA, FBSA, VBA	Blocked, print format

- The Server FTP does not support storage or retrieval of data sets with the T (record overflow) or M (machine carriage control) attributes. However, it does support VBM in binary mode.
- These unusual and unblocked print format RECFMs can be read but not written by the FTP Server:

  FA, FSA, VA, VSA, UA
- When the TYPE is I or L (binary data), the data set RECFM cannot specify print format (A). A print data set can be created or retrieved only as text, not as binary data.

Note: Violation of any of these rules results in a 554 illegal recfm error reply.

Data Set Attribute Errors

These conflicting data set attributes create the following error reply and prevent the operation:

```
501 LRECL or BLKSIZE invalid or inconsistent
```

- RECFM ("V.."): BLKSIZE < 4
- RECFM ("VB" or "VBA"): LRECL > BLKSIZE-4 or LRECL < 8
- RECFM ("FB"): LRECL > BLKSIZE or BLKSIZE not an integer multiple of LRECL

In addition, for unblocked RECFMs the LRECL is forced to do the following:

- RECFM ("V"): BLKSIZE-4
- RECFM ("F"): BLKSIZE

Appending to Empty Data Sets

These rules apply to append (APPE) operations performed to empty data sets.

- If the data set to be appended to has RECFM=0 or BLKSIZE=0, it is assumed to be empty and is scratched and recreated using the new attributes.
- If the data set is not empty but has LRECL=0 and blocked RECFM, Server FTP uses the new LRECL (the default) or the LRECL from SITE.
- Otherwise, Server FTP uses DCB parameters of the existing data set.

JES Internal Reader Support Procedures

These rules apply to the Server FTP SUBMIT operation:

- Connect and log on to the host where the JCL resides.
- Connect and log on to your IBM host where Unicenter TCPaccess FTP Server is running.
- Use a QUOTE type command to send a SITE SUBMIT command to the IBM host (such as, QUOTE SITE SUBMIT).
- Issue a PUT, GET, or SEND command of the data set on the host where the JCL resides to any name on the IBM host (such as, PUT IEBPTPCH *any_name*). The JCL can reside in a physical sequential data set or as a member of a partitioned data set. The new name (*any_name*) on the IBM host is ignored and Unicenter TCPaccess FTP Server submits the job to the JES internal reader.

Defaults

The Server FTP defaults to RECFM=FB,LRECL=80,BLKSIZE=20000.

The defaults can be overridden by the Unicenter TCPaccess FTP Server administrator using the special GAT TYPE(INTRDR) entry.

Usage Guidelines

- The DCB attributes used by Server FTP to allocate the JES internal reader can be explicitly set by the user with the Server FTP SITE IRRECFM, IRLRECL, and IRBLKSIZE commands.
- Server FTP does a minimum of validity checking for internal reader file attributes. Be sure that the attributes selected are compatible with each other and are appropriate for the local Job Entry Subsystem.

Examples

The following examples show JCL streams that can be submitted using SITE SUBMIT to cause the printing of the data included in the submitted job:

```
//IEBPTPCH JOBCARD
//*
//*     IF THIS JOB STREAM RESIDES ON YOUR REMOTE HOST, YOU CAN
//*     SUBMIT THIS JOB TO THE JES INTERNAL READER OF
//*
//*     YOUR IBM SYSTEM BY ENTERING THE FOLLOWING COMMANDS:
//*
//*     FTP MVS * CONNECT TO THE IBM HOST
//*     USERID/PSW * LOGIN TO THE IBM HOST
//*     QUOTE SITE SUBMIT * NOTIFY SNS/TCP SFTP TO
//*     PUT IEBPTPCH * SUBMIT JCL TO JES INTERNAL READER
//PTPCH       EXEC PGM=IEBPTPCH
//SYSPRINT    DD SYSOUT=A
//SYSIN       DD   *
  PRINT PREFORM=FBA
//SYSUT2      DD SYSOUT=*
//     *SYSUT1 CAN BE PS OR MEMBER OF A PDS THAT RESIDES ON THE
//     *MVS SYSTEM
//     *ONE CAN INCLUDE THE PRINT FILE AS INSTREAM DATA AS
//     *SHOWN IN THIS EXAMPLE
//*
//SYSUT1      DD DATA
  (include print file here)
//IEBGENER JOBCARD
//*     THIS JOB STREAM RESIDES ON YOUR REMOTE HOST, YOU CAN
//*     SUBMIT THIS JOB TO THE JES INTERNAL READER OF YOUR IBM
//*     SYSTEM BY ENTERING THE FOLLOWING COMMANDS:
//*
//*     FTP MVS  * CONNECT TO THE IBM HOST
//*     USERID/PSW  * LOGIN TO THE IBM HOST
//*     QUOTE SITE SUBMIT  * NOTIFY SNS/TCP SFTP TO
//*     PUT IEBGENER  * SUBMIT JCL TO JES INTERNAL READER
//GENER       EXEC PGM=IEBGENER
//SYSPRINT    DD SYSOUT=*
//SYSIN       DD DUMMY
//SYSUT2      DD SYSOUT=*
//*     SYSUT1 CAN BE PS OR MEMBER OF A PDS THAT RESIDES ON THE
//*     MVS SYSTEM. ONE CAN INCLUDE THE PRINT FILE AS INSTREAM DATA
//*     AS SHOWN IN THIS EXAMPLE
//*
//SYSUT1       DD DATA
  (include print file here)
```

Data Transfer Operations

This section describes the operation of data transfers by the Server FTP.

Transfer Commands

The following table lists options invoked by the PUT, APPE, and GET commands that initiate data transfer:

Option	Function
STOR	Saves a file on a z/OS and OS/390 system
APPE	Appends to a file stored on a z/OS and OS/390 system
RETR	Retrieves a file from a z/OS and OS/390 system

Transfer Options

A 226-Transfer complete reply is sent only when the disk file being written has been closed successfully or when the data being retrieved and sent across your network has been fully acknowledged.

Truncating and Folding Records

The Server FTP generally folds rather than truncates a network record that exceeds LRECL (or BLKSIZE, for an unblocked data set). You can force truncation with an ALLO R command.

Raveled Files

A *raveled* file is a file that contains the network data concatenated into the file with no record markers. Such a file can be created and later retrieved with FTP without loss of information. However, it is not usually possible to process it with any IBM utility. The only transformation generally done on a raveled file is to translate between ASCII and EBCDIC for TYPE A.

Create or retrieve a raveled file with the FTP parameters STRU F and one of these:

- Character type (TYPE A or TYPE E) and unblocked logical records
- TYPE I or TYPE L

When a raveled file is stored by the Server FTP, it is folded into maximum-sized logical records.

Padding Fixed-length Records

If the data set has a RECFM containing F (that is, it has fixed-length logical records), and if it is not raveled, the Server FTP performs these steps:

- Pads each record being written to the next LRECL
- Removes trailing pad characters from each record read from disk

The pad character is normally blank for character types and zero for binary types. However, the SITE command PAD option can set it to a different value. Although this is not strictly invertible, in most cases it saves transmission time and network bandwidth. The statistics at the end of a file transfer contain the number of records padded if this number is nonzero.

Translation

For the ASCII transfer (TYPE AN, TYPE AT, or TYPE AC), EBCDIC data in a disk file is translated to ASCII over the network and vice versa. The Server FTP always stores data on disk in EBCDIC.

Line Image Files

A line image file is defined by FTP parameters STRU F, one of the character types (TYPE AN, TYPE EN, TYPE AT, or TYPE ET), and a blocked disk data set (i.e., not raveled). With line image files, end-of-line (EOL) in the network data is mapped to z/OS and OS/390 end-of-record and vice versa. Network end-of-line is normally NL (EBCDIC) or CRLF (ASCII). However, CRLF is also recognized in EBCDIC network data. Isolated CR or LF characters are not recognized as an EOL sequence.

Record Structured Files

For the FTP parameter STRU R, a network record is mapped into a z/OS and OS/390 logical record and vice versa.

If ALLO R is specified, each network record that exceeds ALLO R is truncated.

With STRU R, a storage operation (that is, STOR or APPE) that creates a print file (TYPE AT, TYPE AC, TYPE ET, or TYPE EC) requires the data set to be blocked.

If the data set is not blocked, the operation fails with the reply:

```
501 print type and STRU R requires blocked data set.
```

Tabs

Horizontal tab characters (HT) received from the network are expanded or deleted in accordance with the current SITE command TABS option setting. Default is a tab every eight columns; this can be overridden by the SITE TABS command.

A vertical tab (VT) character is always treated as data and stored in the file.

Carriage Control and Format Effectors

For character types with formats N or T, it might be necessary to translate between ASCII format effectors and ASA carriage control. The Server FTP uses simple locally applied rules that do no require buffering data. These rules are invertible between storing and retrieving data with the same parameters.

Some obscure cases cannot be handled correctly in this way. In fact, to fully define the correspondence, you must compare the effects of the files on assumed printing mechanisms to achieve the same appearance. This requires that FTP define additional attributes (such as the size of a page and the effect of an overstrike).

The rules assume that at the beginning of a file, the ASA line printer is positioned on the first line. Therefore, the default carriage control character used to create the first line of an ASA print file is **+** (suppress space), with following lines normally using a blank (single space) before printing.

Character-Type Rules

This section describes the transformation rules for creating and retrieving character-type (TYPE A or TYPE E) files.

In all cases, the network data can be ASCII or EBCDIC, but a disk data set is always EBCDIC. In addition, an HT character in a received file being stored (via STOR or APPE) is always handled in accordance with the current TABS value.

File Structure with No Format Control

The STRU F command with TYPE AN or TYPE EN parameters define a file-structured (line-image) character file with no format control. Generally, with these parameters, network lines are mapped to and from logical records. The network data can be parsed into a series of lines, using the following syntax:

```
text...eol
```

ext	A (possibly null) block of text.
eol	An end-of-line sequence (CRLF or NL).
Notes	Other format effectors, including isolated CR and LF characters, can be included in text.

Line Image Files

If the data set to be written or retrieved is blocked (RECFM includes B), the file is assumed to contain line images.

Storing Line Image Files

These rules apply to line image files being stored (via STOR or APPE):

- An ASCII file (TYPE A) is translated to EBCDIC.
- The text is scanned for the EOL sequence (CRLF or NL).
- Each line is mapped into a z/OS and OS/390 logical record, and the EOL sequence is discarded.
- A line that exceeds the target file LRECL is folded into subsequent logical record(s) rather than being truncated.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.

- If the data set has fixed length records (RECFM includes F), each line is padded with blanks (or the PAD character specified via the SITE command) to the next logical record boundary.
- Any control characters (including format effectors but not including the EOL sequence) are left in the data stream.
- If the data set has ASA carriage control (RECFM includes A), a blank Carriage Control Character (CCC) is inserted at the beginning of each logical record.

Retrieving Line Image Files

These rules apply to line image files being retrieved (via RETR):

- Each z/OS and OS/390 logical record is mapped into a line with an EOL sequence inserted after each line.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has fixed-length logical records (RECFM includes F), all trailing blanks (or the PAD character specified via the SITE command) are stripped off.
- If the data set has ASA carriage control (RECFM includes A), an additional transformation is applied. If the ASA CCC is the first character of the logical record and *text* is the rest, the data shown in the following table is sent over the network:

CCC	Network data
blank	eol *text*
0	eol eol *text*
-	eol eol eol *text*
1	CR FF *text*
+ (first record)	*text*
+ (other)	CR *text*

A single logical record can result in a series of network lines, and a **+** CCC in the first record of the file is effectively ignored.

Raveled Files

If the data set to be written or retrieved is unblocked (RECFM does not include B), the file is assumed to be a raveled file and is treated as a single-byte string.

Storing Raveled Files

The rules in this list apply to a raveled file being stored (via STOR or APPE).

The data set stored on disk is a concatenation of the network data. That is, the network data is folded into logical records.

An ASCII file (TYPE A) is translated to EBCDIC.

Retrieving Raveled Files

These rules apply to a raveled file being retrieved (via RETR):

- The network data set is a concatenation of the retrieved logical records. That is, the logical records retrieved from disk are raveled into a single stream of bytes for transmission across the network.
- If the data set has ASA carriage control (RECFM includes A), the CCC is deleted and is not included in the network data.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.

File Structure with Telnet Format

The STRU F command with TYPE AT or TYPE ET parameters define a file-structured character file with Telnet format effectors. Generally, with these parameters, a network line maps to/from a logical record.

Network data can be parsed into a series of segments using the following format:

```
fe  text
```

fe	A sequence of ASCII format effectors.
text	A (possibly null) block of text.

Notes

Data storage with these parameters follows the same rules as storage of STRU F, TYPE AN or TYPE EN, with one exception.

In the line image case, if the data set being stored into (via STOR or APPE) has ASA carriage control (RECFM includes A), the CCC is not set to blank (as in the STRU F, TYPE AN or TYPE EN case) but is determined from the network data according to the table in Retrieving Line Image Files. Any format effectors not involved in this transformation are left in the data stream.

Retrieval of data using these parameters follows the same rules as retrieval of STRU F, TYPE AN, or TYPE EN.

File Structure with ASA Format

The STRU F command with TYPE AC or TYPE EC parameters define a file-structured (line-image) print file containing ASA carriage control. Generally, with these parameters, a network line maps to/from a logical record. The network data consists of these parameters:

```
cc text...eol
```

cc	An ASA Carriage Control Character.
ext	A (possibly null) block of text.
eol	An end-of-line sequence (CR, LF, or NL).

Line Image Files

With these parameters, if the data set to be written or retrieved is blocked (RECFM includes B), the file is considered to be line images.

Storing Line Image Files

These rules apply to line image files being stored (via STOR or APPE):

- An ASCII file (TYPE A) is translated to EBCDIC.
- The text is scanned for the EOL sequence (CRLF or NL).
- Each line is mapped into a logical record, and the EOL sequence is discarded.
- A line that exceeds the target file LRECL is folded into subsequent logical record(s), and a warning message is issued. If the file being written has ASA carriage control, each of the subsequent logical records is stored with a blank CCC.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- If the data set has fixed length records (RECFM includes F), each line is padded with blanks (or the PAD character specified via the SITE command) to the next logical record boundary.
- Any control characters (including format effectors but not including the EOL sequence) are left in the data stream.
- If the data set has ASA carriage control (RECFM includes A), the ASA CCC from the network data is used as the CCC for the stored logical record. If the data set does not have ASA carriage control, the first character of the network record is not stored in the logical record.

The Server FTP writes such a print file only into a blocked data set (RECFM includes B).

Retrieving Line Image Files

When a RETR operation is performed with these TYPE and STRU parameters from a blocked data set, a new line image file is created and sent over the network. These rules apply to line image files being retrieved (via RETR):

- Each z/OS and OS/390 logical record is mapped into a line with an EOL sequence inserted after each line.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has fixed length records (RECFM includes F), all trailing blanks (or the PAD character specified on the SITE command) are stripped off before sending on the network.
- If the retrieved data set contains ASA carriage control (RECFM includes A), the CCC from the logical record is passed as part of the network record. Otherwise, a blank CCC is inserted at the beginning of each network record, except for the first network record into which a **+** CCC is inserted.

Raveled Files

If the data set to be written or retrieved is unblocked (RECFM does not include B), the file is considered to be a raveled file and is treated as a single string of bytes.

Storing Raveled Files

The Server FTP does not write a raveled (unblocked) file with these data transfer parameters.

Retrieving Raveled Files

These rules apply to a raveled file being retrieved (via RETR):

- The network data set is a concatenation of the retrieved logical records. That is, the logical records retrieved from disk are raveled into a single stream of bytes for transmission across the network.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the data set has ASA carriage control (RECFM includes A), the CCC of the first logical record is included in the network data. Any other carriage control is deleted. If the data set does not have carriage control, a **+** CCC is inserted at the beginning of the network data, and no other carriage control is inserted.

Record Structure with No Format

The STRU R command with TYPE AN, TYPE EN, TYPE AT, or TYPE ET parameters define a character file with record structure. Generally, network records are mapped to/from logical records.

```
fe text..eor
```

fe	A (possibly null) sequence of ASCII format effectors.
text	A (possibly null) block of text.
eor	An end-of-record sequence.

Storing Logical Records

These rules apply to files being stored (using STOR or APPE) with these parameters:

- An ASCII file (TYPE A) is translated to EBCDIC.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- A network record longer than ALLO R (if specified) is truncated, and a warning message is issued.
- If a network record (after possible ALLO R truncation) is longer than LRECL, it is folded into subsequent logical record(s) and a warning message is issued.
- If the data set has fixed length logical records (RECFM includes F), each network record is padded with blanks (or the PAD character specified via the SITE command) to the next logical record boundary.
- If the data set has ASA carriage control (RECFM includes A), the CCC is normally blank, and any embedded ASCII format effectors are treated as data. However, if the TYPE is AT or ET, the escape sequences shown in Retrieving Logical Records (F and R) are scanned for and used to set CCC if found in the network data. No scanning for the escape sequences occurs with TYPE AN or TYPE EN.

The Server FTP writes a print file with Telnet format effectors (TYPE AT or TYPE ET) only into a blocked data set (RECFM includes B).

Retrieving Logical Records

These rules apply to files being retrieved (via RETR) with these parameters:

- Each logical record (RECFM includes B) or each physical record (RECFM does not include B) is mapped into a network record. If the data set is unblocked (raveled), the network data is a concatenation of the logical records. The other transformations described in the following steps still apply.
- If the data set has fixed length logical records (RECFM includes F), all trailing blanks (or the PAD character specified via the SITE command) are stripped off before data is sent on the network. This occurs even for unblocked data sets. A completely blank (or null) record is sent as a null record with the appropriate end of record sequences based on mode.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.

- If the data set has ASA carriage control (RECFM includes A), an additional transformation is applied. The following table shows the transformations that map the CCC in the disk data set into network data. A one-to-many relationship can occur between disk records and network records.

CCC	Network Data
blank	*text eo***r**
0	*eor text eor*
-	*eor eor text eor*
1	F' *text eor*
+ (first)	R' *text eor*
+ (other)	*text eor*

Note: If the type is ASCII (TYPE AN or TYPE AT), the EBCDIC single quote (') escape character is translated to an ASCII backslash (\).

Record Structure with ASA Format

The STRU R command with TYPE AC or TYPE EC parameters define a record-structured print file containing ASA carriage control. Generally, with these parameters, network records map to/from logical records.

```
cc text...eor
```

cc	An ASA CCC.
text	A (possibly null) block of text.
eor	An end-of-record sequence.

Storing Print Files

These rules apply to files being stored (via **ST**OR or APPE) with these parameters:

- An ASCII file (TYPE A) is translated to EBCDIC.
- A null record is stored as all pad characters if the RECFM includes F, zero data bytes if the RECFM includes V, or x'00' if the RECFM is U.
- A network record longer than ALLO R (if specified) is truncated, and a warning message is issued.
- If a network record (after possible ALLO R truncation) is longer than LRECL, it is folded into subsequent logical record(s) and a warning message is issued. If the file being written has ASA carriage control (RECFM includes A), each of the subsequent logical records is stored with a blank CCC.
- If the data set has fixed length logical records (RECFM includes F), each network record is padded with blanks (or the PAD character specified via the SITE command) to the next logical record boundary.
- If the data set has ASA carriage control (RECFM includes A), the ASA CCC from the network data (cc) is used as the CCC for the stored logical record. Otherwise, the first character of the network data is deleted and is not stored in the logical record.

The Server FTP writes such a print file only into a blocked data set (RECFM contains B).

Retrieving Print Files

These rules apply to files being retrieved (via RETR) with these parameters:

- Each logical record (RECFM includes B) or each physical record (RECFM does not include B) is mapped into a network record. If the data set is unblocked (raveled), the network data set is a concatenation of the logical records. The other transformations described in the following steps still apply.
- If the data set has fixed length logical records (RECFM includes F), all trailing blanks (or the PAD character specified via the SITE command) are stripped off before sending on the network. This occurs even for unblocked data sets. A completely blank (null) record is sent as a null record with the appropriate end of record sequences based on mode.
- If the type is ASCII (TYPE A), the data is translated from EBCDIC to ASCII.
- If the retrieved data set has ASA carriage control (RECFM includes A), the CCC from the logical record is passed as part of the network record. Otherwise, a blank CCC is inserted at the beginning of each network record, except for the first record, into which a **+** CCC is inserted.

Binary-Type Rules

This section describes the transformation rules for creating and retrieving binary-type (TYPE I or TYPE L) files. The term *image-type* is used interchangeably with *binary-type* in the information in this section.

No Record Structure

The STRU F command with TYPE I or TYPE L parameters defines a binary file without record structure.

Storing Binary Files

These rules define the transformations to such a file when it is stored (using STOR or APPE) by the Server FTP:

- The received network data is folded into maximum-length logical records when stored by the Server FTP.
- If a null (zero-length) network record is received, it is ignored.
- No translation, truncation, or padding is performed on the data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not write image-type data into a print data set (RECFM includes A).

Retrieving Binary Files

These rules define the transformations to such a file when it is retrieved (using RETR) by the Server FTP:

- The retrieved logical records are concatenated into a single stream of network data.
- No translation, truncation, or removal of padding is performed on the retrieved data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not retrieve from a print data set (RECFM includes A).

Record Structure

The STRU R command with TYPE I or TYPE L parameters defines a binary file with record structure.

Storing Structured Binary Files

These rules define the transformations to such a file when it is stored (using STOR or APPE) by the Server FTP:

- If a network record with zero data bytes is received, it is stored as all pad characters if RECFM includes F, zero data bytes if RECFM includes V, and x'00' if RECFM is U.
- A network record longer than ALLO R (if specified) is truncated to ALLO R, and a warning message is issued.
- If a record (possibly truncated to ALLO R) is longer than the maximum logical record length, it is folded into multiple logical records, and a warning message is issued.
- If the data set has fixed-length logical records (RECFM includes F), each network record is padded with zeros (or the PAD character specified via the SITE command) to the next logical record boundary.
- No translation is performed on the data.

Since the contents of an image-type file are considered to be untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not write image-type data into a print data set (RECFM includes A).

Retrieving Structured Binary Files

These rules define the transformations performed on such a file when retrieved (via RETR) by the Server FTP:

- The retrieved logical records are mapped into network records.
- If the data set has fixed length records (RECFM includes F), all trailing zero bytes (or the PAD character specified via the SITE command) are stripped off before sending on the network.
- No translation is performed on the retrieved data.

Since the contents of an image-type file are considered untranslatable as characters, it is assumed not to be a print file. Therefore, the Server FTP does not retrieve from a print data set (RECFM includes A) for transmission as image-type data.

Non-Invertible Retrieval

Unless a file is raveled, storing it with the Server FTP might transform the data in ways that are not strictly invertible. (That is, if the file is later retrieved using the same parameters, it might not be identical to the original file sent to this Server FTP.) The file would generally mean the same thing, but some of its byte stream is changed.

Sources of Non-Invertibility

Here are some of the sources of non-invertibility introduced when a file is stored by this Server FTP:

- All files

  If the data is stored into a data set with fixed-length blocks and if a record/line ends with the pad character the Server FTP uses to pad the block, that pad character is stripped off when the file is retrieved.

- Character files

  HTs might be expanded to blanks during storage and not be reintroduced when the file is retrieved.

  LF and FF, which do not appear at the left margin, insert blanks to position the virtual print head correctly. On RETR, these blanks and the extra CRLF that were not in the original input are returned to the remote host.

  NL is used interchangeably with CRLF in an EBCDIC file. Therefore, CRLF received in such a file is sent out as an NL.

  LFs preceding an FF with no intervening data are removed.

Other Features

This section lists some other useful features of the FTP program.

Testing and Debugging

The STAT display includes the Internals section that contains a number of important control fields and pointers. This information is generally of interest only to the system programmer.

The TCP-level tracing and debugging facilities are described in the *Administrator Guide*.

Telnet Break

The Telnet commands IP, BRK, or CONTROL-C on the control connection stops output from commands such as HELP and STATUS.

Server FTP JES Support

Unicenter TCPaccess FTP Server supports batch FTP for JES in Server FTP. Server FTP offers JES spool support compatible with that offered by the IBM C FTP server. It offers the same functionality as the IBM server when the file type is defined as JES.

The Server FTP JES interface allows users to submit, display, retrieve, and delete a *user's job* and its spool output. *User's job* is defined as those jobs that are assigned an OWNERID or job name equal to the FTP user's user ID plus one character (depending on the JESFILTER option used in the APPCFG*xx*).

Using the Server FTP JES Interface

The Server FTP JES interface allows users to submit, display, retrieve and delete a *user's job* and its spool output. *User's job* is defined as those jobs that are assigned an OWNERID or job name equal to the FTP user's user ID plus one character (depending on the JESFILTER option used in the T051CFA*x*).

Submitting User's Job from a Client

To submit a user’s job from a client, complete the following steps:

1. Using an editor, create the JCL and data you wish to submit.
2. Establish a session with the FTP server on the z/OS and OS/390 system to which you want to submit.
3. Enter the FTP JES interface mode by entering:

   QUOTE SITE FILETYPE=JES
4. Submit the created JCL file by entering:

   PUT ***jclFile***
5. The job is submitted to the JES internal reader under the user ID you used when you logged on to the FTP server.
6. To exit FTP JES mode (back to normal FTP mode), enter:

   QUOTE SITE FILETYPE=SEQ

Note: The FTP JES interface mode stays on until you issue this command or terminate the FTP session.

Displaying the Status of a Job

You can display the status of user's job with the FTP subcommands DIR or LIST while in FILETYPE=JES mode. Output from DIR subcommand might look like the following:

```
USERA1   JOB03841 OUTPUT 3 spool Files
USERA3   JOB03907 OUTPUT 5 spool Files
USERA2   JOB03855 ACTIVE
USERA4   JOB03923 INPUT
```

The output is formatted as follows:

- First column—job name.
- Second column—job ID assigned by JES.
- Remainder of display shows user's job status:

INPUT	User's job received but has not run yet.
ACTIVE	User's job is running.
OUTPUT	User's job is completed. You also see the number of held spool files created which could be retrieved.

Note: If a user's job was submitted with a held MSGCLASS, the order of the spool files is JCL messages, JES messages, z/OS and OS/390 initiator/terminator messages, followed by the held SYSOUT files.

Output from the LIST subcommand (using the same example) looks like the following:

```
JOB03841
JOB03907
JOB03855
JOB03923
```

Though LIST may not provide desired information, it uses less computer resources and enables client FTPs that support the MGET subcommand.

Retrieving User's Job Spool Files

To retrieve all held spool files created by a user's job, enter FTP JES interface mode (FILEtype=JES), and then specify one of the following:

GET *jobid* [*outputFile*]

GET *jobid.x* [*outputFile*] (for compatibility with IBM FTP)

All the held spool files are transferred and placed into *client.filename* (if specified) or *jobid* (or *jobid.x*). The following line appears after each retrieved JES spool file to allow you to find the end of each file:

```
!! END OF JES SPOOL FILE !!
```

For example:

```
GET JOB04197.X
```

Retrieves all held spool files for job ID JOB04197 and places them into a file named JOB04197.X (prefixed by local directory).

To retrieve only certain spool files, specify:

```
GET jobid.# [outputFile]
```

Replace the # with the file number you desire and <*client.filename*> (optional) is replaced by the client file name to receive the data.

For example:

```
GET JOB04197.1 USERA.TESTJOB
```

Gets the first held spool file of job ID JOB04197 and places it into USERA.TESTJOB (prefixed by local directory).

Note: In JES3, the spool files must be in a hold queue reserved for external writers (HOLD=EXTWTR).

Deleting a User's Job

You can delete a user's job before or during execution, or delete the output after the user's job completed. To do so, enter the FTP JES interface mode and use the FTP DELete subcommand.

For example:

```
DELETE JOB04197
```

The host returns the message 220 CANCEL SUCCESSFUL after it deletes the job.

Note: When you issue the DELete subcommand all spool output related to the user's job is deleted—both held and non-held spool.

Submitting User's Job and Automatically Retrieving Output (PUTGET)

You can submit a user's job using FTP and automatically retrieve its held output. This function requires the JCL to be built on the FTP server site. The following FTP commands are used:

```
QUOTE SITE FILETYPE=JES
GET jclFile outputFile
```

jclFile	Defines the data set on the server site which contains the JCL for the user's job.
outputFile	Defines the data set on the client site that is to contain the held spool output when the job completes.

The MVS server reads *jclFile* and submits it to the JES internal reader. It waits for the submitted user's job to complete. It then retrieves all the held spool files and sends them to the client.

When using this function, remember that your session is suspended till the user's job completes or JESPUTGETTO time limit is reached (defined in APPCFG*xx*). If a timeout occurs, you will have to manually retrieve your output as described in the section, Retrieving User's Job Spool Files.

Examples

The following examples were performed with JESFILTER(OWNERID) set in the active APPCFG*xx*:

<-- Enter FTP JES interface mode -->

```
ftp> quote site file=jes        <-- Can abbreviate FILEtype
200 OK, Ready
```

<-- Check JES queue for user's jobs -->

```
ftp> dir
200 OK, Ready
125 Transfer started
No jobs found on JES queue
226-Transfer complete.  28 bytes sent in 0.13 seconds (215 bytes/s)
    User=ABC1  Data bytes read: 26.
226 End of reply.
28 bytes received in 0.16 seconds (0.17 Kbytes/s)
```

<-- Submit job from client -->

```
ftp> put sample.jcl
200 OK, Ready
150-Dataset opened; data connection starting.
    Data transfer Type is ASCII. Structure is File. Mode is Stream.
    Recfm=FB  Lrecl=80  Blksize=80
    Data will be written to the JES internal reader.
150 Network data which exceeds LRECL will be wrapped to the next record.
226-Transfer complete.  1929 bytes received in 0.27 seconds (7144 bytes/s)
    Submitted job JOB06327  User=ABC1  Data bytes written: 3120.
226 Records padded: 39
1929 bytes sent in 0.02 seconds (96.45 Kbytes/sec)
```

<-- Check status of submit -->

```
ftp> dir
200 OK, Ready
125 Transfer started
ABC1J     JOB06327  ACTIVE
226-Transfer complete.  28 bytes sent in 0.11 seconds (254 bytes/s)
    User=ABC1  Data bytes read: 26.
226 End of reply.
28 bytes received in 0.24 seconds (0.12 Kbytes/s)
```

<-- Recheck the status -->

```
ftp> dir
200 OK, Ready
125 Transfer started
ABC1J     JOB06327  OUTPUT    5 Spool Files
226-Transfer complete.  45 bytes sent in 0.13 seconds (346 bytes/s)
    User=ABC1  Data bytes read: 43.
226 End of reply.
45 bytes received in 0.16 seconds (0.28 Kbytes/s)
```

<-- Send 1st file to client -->

```
ftp> get j6327.1 sample.file1
200 OK, Ready
150-Dataset opened; data connection starting.
    Data transfer Type is ASCII. Structure is File. Mode is Stream.
150 Dataset name: ABC1.ABC1J.JOB06327.D0000002.JESMSGLG
226-Transfer complete.  1293 bytes sent in 0.19 seconds (6805 bytes/s)
    Dataset name: ABC1.ABC1J.JOB06327.D0000002.JESMSGLG  User=ABC1
    Data bytes read: 1964.
226 End of reply.
local: sample.file1 remote: j6327.1
1293 bytes received in 0.025 seconds (50 Kbytes/s)
```

<-- Send all held sysout to client -->

```
ftp> get j6327 sample.allfiles
200 OK, Ready
150-Dataset opened; data connection starting.
    Data transfer Type is ASCII. Structure is File. Mode is Stream.
150 Sending all spool files for JOB06327
226-Transfer complete.  724615 bytes sent in 9.13 seconds (79366 bytes/s)
    User=ABC1  Data bytes read: 866617.
226 End of reply.
local: sample.allfiles remote: j6327

724615 bytes received in 9 seconds (79 Kbytes/s)
```

<-- Submit from server & send results to client (PUTGET) -->

```
ftp> get sample.cntl(iehlist) iehlist.allfiles
200 OK, Ready
150-Submitting job ABC1.SAMPLE.CNTL(IEHLIST) FIXlrecl 80
    When JOB06349 is done, will retrieve its output
    Dataset opened; data connection starting.
    Data transfer Type is ASCII. Structure is File. Mode is Stream.
150 Sending all spool files for JOB06349
226-Transfer complete.  34641 bytes sent in 1.13 seconds (30655 bytes/s)
    User=ABC1  Data bytes read: 37124.
226 Disk tracks read: 1.
local: iehlist.allfiles remote: sample.cntl(iehlist)
34641 bytes received in 1.1 seconds (30 Kbytes/s)
```

<-- Check the JES queue for user's jobs -->

```
ftp> dir
200 OK, Ready
125 Transfer started
ABC1VTOC  JOB06349  OUTPUT    4 Spool Files
ABC1J     JOB06327  OUTPUT    5 Spool Files
226-Transfer complete.  90 bytes sent in 0.12 seconds (750 bytes/s)
    User=ABC1  Data bytes read: 86.
226 End of reply.
90 bytes received in 0.17 seconds (0.51 Kbytes/s)
```

<-- Send all user's jobs held sysout to client -->

```
ftp> mget all
mget JOB06349? y
200 OK, Ready
150-Dataset opened; data connection starting.
    Data transfer Type is ASCII. Structure is File. Mode is Stream.
150 Sending all spool files for JOB06349
226-Transfer complete.  34641 bytes sent in 1.13 seconds (30655 bytes/s)
    User=ABC1  Data bytes read: 36404.
226 End of reply.
local: JOB06349 remote: JOB06349
34641 bytes received in 0.97 seconds (35 Kbytes/s)
mget JOB06327? y
200 OK, Ready
150-Dataset opened; data connection starting.
    Data transfer Type is ASCII. Structure is File. Mode is Stream.
150 Sending all spool files for JOB06327
226-Transfer complete.  724615 bytes sent in 9.15 seconds (79192 bytes/s)
    User=ABC1  Data bytes read: 866617.
226 End of reply.
local: JOB06327 remote: JOB06327

724615 bytes received in 9 seconds (79 Kbytes/s)
```

<-- Delete JOB06327 (ABC1J) -->

```
ftp> del j6327
250 Job cancelled OK
```

<-- Check JES queue again -->

```
200 OK, Ready
125 Transfer started
ABC1VTOC  JOB06349  OUTPUT    4 Spool Files
226-Transfer complete.  45 bytes sent in 0.13 seconds (346 bytes/s)
    User=ABC1  Data bytes read: 43.
226 End of reply.

45 bytes received in 0.17 seconds (0.26 Kbytes/s)
```

<-- Exit FTP/JES interface mode -->

```
ftp> quote site file=seq
200 OK, Ready
```

Chapter 4

Load Module Transfer (LMTR)

Load module transfer enable MVS load modules to be transferred between load libraries on different hosts or the same host. The FTP clients and servers participating in the load module transfer can be either Unicenter TCPaccess FTP Server components or IBM Communications Server for z/OS and OS/390 FTP Server components.

Important! *For load module transfer to be successful, all Unicenter TCPaccess FTP Server components must be at Release 6.0 SP1 or higher and all IBM Communications Server for z/OS and OS/390 FTP Server components must be at release V2R10 or higher.*

Specifying a LMTR

A load module can be specified by its real or alias name; in either case the real module and all its associated aliases are transferred. The FTP commands GET, MGET, PUT, and MPUT properly transfer MVS load modules. After a successful FTP transfer, the load module(s) are executable on the destination system.

The load module transfer function uses the IEBCOPY system utility, which must be available on both the originating and destination systems. On the origin system, the load module(s) to be transferred are unloaded to a temporary data set, which is then transferred to the destination system and reloaded. Sufficient temporary DASD to hold the unload file must be available on both the originating and destination systems.

Restrictions

Due to the special requirements of MVS load modules, there are some restrictions:

- The current working directory on both the local and remote host must be the source and destination load library. A load library is a PDS or PDSE with RECFM=U.
- Only member names can be specified. No fully qualified names may be specified.
- File rename is not supported on load module transfer.
- Load modules can only be transferred between the same types of libraries. For example, PDS to PDSE transfer is not allowed.
- A load module loading from a temporary data set will always be a REPLACE operation overwriting existing members.
- There is not prompting on MPUT and MGET commands. All files that match the mask provided are transferred.

In most cases where load module processing cannot be performed, including failure to abide by the restrictions given above, FTP completes the file transfer using normal processing. Any load modules transferred with normal processing will not be executable on the target system. Some reasons for failure of load module transfer but successful normal transfer processing include:

- Not all hosts implement the load module transfer protocol
- Attempt to rename a file or specify it as anything other than a member name
- Attempt to transfer from a PDSE to a PDS or any other non-PDSE file structure

Note: When a cd or lcd subcommand is processed, a message or reply is issued to notify the FTP user if the directory is eligible for load module transfer processing. If this message or reply is not issued when changing to a directory, load module transfer is not attempted.

Index