



IBM Systems - iSeries  
Networking  
File Transfer Protocol

*Version 5 Release 4*







IBM Systems - iSeries  
Networking  
File Transfer Protocol

*Version 5 Release 4*

**Note**

Before using this information and the product it supports, read the information in "Notices," on page 153.

**Seventh Edition (February 2006)**

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

© Copyright International Business Machines Corporation 1998, 2006. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>File Transfer Protocol . . . . .</b>	<b>1</b>
Printable PDF . . . . .	1
Scenarios: File Transfer Protocol . . . . .	1
Scenario: Transfer a file from a remote host . . . . .	1
Scenario: Secure File Transfer Protocol with Secure Sockets Layer . . . . .	3
Configure the File Transfer Protocol server . . . . .	7
File Transfer Protocol server in iSeries Navigator . . . . .	7
Configure FTP servers for graphical FTP clients and Web tools . . . . .	8
Configure anonymous File Transfer Protocol . . . . .	10
Secure File Transfer Protocol. . . . .	14
Prevent File Transfer Protocol server access. . . . .	15
Control File Transfer Protocol access . . . . .	16
Use Secure Sockets Layer to secure the File Transfer Protocol server . . . . .	17
Secure the FTP client with Transport Layer Security or Secure Socket Layer. . . . .	20
Manage access using File Transfer Protocol exit programs . . . . .	22
Manage access using iSeries Navigator . . . . .	23
Monitor incoming File Transfer Protocol users. . . . .	23
Manage the File Transfer Protocol server . . . . .	24
Start and stop the File Transfer Protocol server . . . . .	24
Set number of available File Transfer Protocol servers . . . . .	25
Improve FTP server performance with configurable subsystem support . . . . .	25
Use the File Transfer Protocol client on iSeries server . . . . .	25
Start and stop a client session . . . . .	26

Server timeout considerations . . . . .	29
Transfer files with File Transfer Protocol. . . . .	29
Run File Transfer Protocol in unattended mode using a batch job . . . . .	30
File Transfer Protocol reference information. . . . .	39
File Transfer Protocol server subcommands. . . . .	39
File Transfer Protocol client subcommands . . . . .	57
File Transfer Protocol exit programs . . . . .	87
Data transfer methods . . . . .	124
File systems and naming conventions . . . . .	133
i5/OS file systems that are supported by File Transfer Protocol . . . . .	134
File Transfer Protocol server reply status messages . . . . .	134
File Transfer Protocol server syntax conventions . . . . .	136
File Transfer Protocol client syntax conventions . . . . .	137
Troubleshoot FTP . . . . .	143
Determine problems with File Transfer Protocol . . . . .	143
Materials required for reporting FTP problems . . . . .	146
Trace the FTP server . . . . .	146
Trace the FTP client . . . . .	149
Work with FTP server jobs and job log . . . . .	150
Code license and disclaimer information . . . . .	151

<b>Appendix. Notices . . . . .</b>	<b>153</b>
Programming Interface Information . . . . .	154
Trademarks . . . . .	155
Terms and conditions. . . . .	155



---

## File Transfer Protocol

You can set up your iSeries™ server to send, receive, and share files across networks by using the file transfer protocol (FTP). You can also rename, add, and delete files. Before you set up your system to transfer files, you must have TCP/IP configured and started on your system.

**Note:** By using the code examples, you agree to the terms of the Code license and disclaimer information.

---

### Printable PDF

Use this to view and print a PDF of this information.


To view or download the PDF version of this document, select FTP (about 1636 KB).

### Saving PDF files

To save a PDF on your workstation for viewing or printing:

1. Right-click the PDF in your browser (right-click the link above).
2. Click the option that saves the PDF locally.
3. Navigate to the directory in which you want to save the PDF.
4. Click **Save**.

### Downloading Adobe Reader

- 1 You need Adobe Reader installed on your system to view or print these PDFs. You can download a free  
1 copy from the Adobe Web site ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html)) .

---

## Scenarios: File Transfer Protocol

Read scenarios to understand how File Transfer Protocol (FTP) is configured and used.

The following scenarios help you understand how FTP works, and how you can use an FTP environment in your network. These scenarios introduce fundamental FTP concepts from which beginners and experienced users can benefit before they proceed to the planning and configuration tasks.

### Scenario: Transfer a file from a remote host

To transfer a file to a test server, use basic File Transfer Protocol (FTP) to send the file to the remote host.

#### Situation

Suppose that a colleague developed Java™ files on a remote server. As a system engineer, you need to transfer the example.jar file from the remote server to your local test server. The client and the server are both an iSeries using i5/OS™ FTP.

#### Objectives

Use FTP to transfer the file across a TCP/IP network.

## Details

To transfer the file, two connections are used: the *control connection* and the *data connection*. The control connection is used to send subcommands from the client to the server and receive responses to those commands from the server to the client. The client initiates FTP commands to the FTP server. The data connection is used to transfer the actual files. Both the client and the server interface with the i5/OS file system.

To transfer files, you need a user ID on both systems. You also need the following system requirements:

- iSeries server running i5/OS
- TCP/IP Connectivity Utilities (5722-TC1)
- FTP server configured

To transfer files, you also need to know the following information:

- Host name of the remote system
- Your user name and password on the remote system
- Name of file to transfer
- Location of the file to transfer
- File format (format that you must transfer the file in, such as binary or ASCII)

### Configuration tasks

You must complete each of these tasks to perform a simple file transfer:

**Note:** You can also transfer files automatically by using FTP as a batch job.

1. Start your FTP client session. For this scenario: In the iSeries character-based interface, type STRTCPFTP and press Enter.
2. Specify the name of the remote system to which you want to send the file.  
For this scenario: theirco.com
3. Tell the remote system your user name for the remote server. For this scenario:  
Enter login ID (yourid):  
===>yourid
4. Tell the remote system your password for the remote server. For this scenario:  
Enter password:  
===>yourpassword
5. Locate the directory on the TheirCo server from which you want to transfer the file. For this scenario:  
===>cd /qibm/userdata/os400/dirserv/usertools/windows
6. Navigate to the directory on the local server to which you want to transfer the file. For this scenario:  
===>lcd /qibm/userdata/os400/dirserv/usertools/windows
7. Specify file type, ASCII or BINARY. Default file type is ASCII. For a .jar file, you must switch the file transfer type to binary. For this scenario: ===> binary
8. Request a file transfer from the remote server system to the client system. For this scenario: ===> get example.jar
9. When finished, Exit from FTP.  
For this scenario: ===> QUIT

#### Related tasks

“Transfer files with File Transfer Protocol” on page 29

You can send and receive files with File Transfer Protocol (FTP).

#### Related reference



“Run File Transfer Protocol in unattended mode using a batch job” on page 30  
This topic provides examples of how to run File Transfer Protocol (FTP) in an unattended mode.  
“Start and stop a client session” on page 26  
You can start and stop a client session from this topic.  
“ASCII (Change File Type to ASCII)” on page 60  
“BINARY (Set Transfer Type to Image)” on page 61

## Scenario: Secure File Transfer Protocol with Secure Sockets Layer

Use Secure Sockets Layer (SSL) to secure data being transferred to your partner company.

### Situation

Suppose that you work for MyCo, a company that researches startup companies and sells the research to companies in the investment planning industry. One such company, TheirCo, needs the services that MyCo provides, and would like to receive research reports through File Transfer Protocol (FTP). MyCo has always ensured the privacy and security of the data it disperses to its customers--whatever the format. In this case, MyCo needs SSL-secured FTP sessions with TheirCo.

### Objectives

The following items are your objectives in this scenario:

- Create and operate a local Certificate Authority on the MyCo iSeries server
- Enable SSL for MyCo's FTP server
- Export a copy of MyCo's local CA certificate to a file
- Create a \*SYSTEM certificate store on TheirCo's server
- Import MyCo's local CA certificate into TheirCo's \*SYSTEM certificate store
- Specify MyCo's local CA as a trusted CA for TheirCo's FTP client

### Prerequisites

#### MyCo

- Has an iSeries server that is running on OS/400® V5R1 or later of i5/OS operating system.
- Has the V5R1 or later TCP/IP Connectivity Utilities (5722-TC1) installed on the iSeries server.
- Has the Cryptographic Access Provider 128-bit for iSeries server (5722-AC3) installed on their iSeries server.
- Has the IBM® Digital Certificate Manager (DCM) (5722-SS1 option 34) installed on the iSeries server.
- Has the IBM HTTP Server (5722-DG1) installed on the iSeries server.
- Uses certificates to protect access to public applications and resources.

#### TheirCo

- Has an iSeries server that is running OS/400 V5R2 or later of the operating system.
- Has the V5R2 TCP/IP Connectivity Utilities (5722-TC1) installed on the iSeries server.
- Has the Cryptographic Access Provider 128-bit for iSeries server (5722-AC3) installed on their iSeries server.
- Has the IBM Digital Certificate Manager (5722-SS1 option 34) installed on the iSeries server.
- Has the IBM HTTP Server (5722-DG1) installed on the iSeries server.
- Uses an i5/OS operating system with a TCP/IP FTP client for FTP sessions.

## Details

TheirCo uses an i5/OS operating system with a FTP client to request a secure FTP file transfer from MyCo's FTP server. The server is authenticated. TheirCo receives financial reports from MyCo by using an SSL-secured FTP session.

### Related concepts

"Secure the FTP client with Transport Layer Security or Secure Socket Layer" on page 20

You can use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) connections to encrypt data transferred over File Transfer Protocol (FTP) control and data connections.

### Related tasks

Use a public certificate

Start DCM

### Related information

Scenario: Use certificates to protect access to public applications and resources

## Configuration details

In order to secure File Transfer Protocol (FTP) with Secure Sockets Layer (SSL), you need to configure the server.

The following tasks are completed by MyCo and TheirCo to secure their FTP sessions with SSL.

### Create and operate a local Certificate Authority on the MyCo iSeries™ server:

This scenario assumes that MyCo has not used Digital Certificate Manager (DCM) previously to set up certificates for its iSeries server. Based on the objectives for this scenario, MyCo has chosen to create and operate a local Certificate Authority (CA) to issue a certificate to the FTP server. However, MyCo can use DCM to configure the File Transfer Protocol (FTP) server to use a public certificate for SSL instead.

When using Digital Certificate Manager (DCM) to create a local CA, you are guided through a process that ensures that you configure everything needed to enable Secure Sockets Layer (SSL).

MyCo uses the following steps to create and operate a local CA on their server, using the Digital Certificate Manager (DCM):

1. Start DCM.
2. In the navigation frame of DCM, select **Create a Certificate Authority (CA)** to display a series of forms. These forms guide you through the process of creating a local CA and completing other tasks needed to begin using digital certificates for SSL, object signing, and signature verification.
3. Complete all the forms that display. There is a form for each of the tasks required to create and operate a local CA on the iSeries server.
  - a. Choose how to store the private key for the local CA certificate. This step is included only if you have an IBM 4758-023 PCI Cryptographic Coprocessor installed on your iSeries server. If your system does not have a cryptographic coprocessor, DCM automatically stores the certificate and its private key in the local CA certificate store.
  - b. Provide identifying information for the local CA.
  - c. Install the local CA certificate on your PC or in your browser. This enables software to recognize the local CA and validate certificates that the CA issues.
  - d. Choose the policy data for your local CA.
  - e. Use the new local CA to issue a server or client certificate that applications can use for SSL connections. If you have an IBM 4758-023 PCI Cryptographic Coprocessor installed in the iSeries server, this step allows you to select how to store the private key for the server or client certificate.

If your system does not have a coprocessor, DCM automatically places the certificate and its private key in the \*SYSTEM certificate store. DCM creates the \*SYSTEM certificate store as part of this task.

- f. Select the applications that can use the server or client certificate for SSL connections.

**Note:** Be sure to select the application ID for the i5/OS TCP/IP FTP server (QIBM\_QTMF\_FTP\_SERVER).

- g. Use the new local CA to issue an object signing certificate that applications can use to digitally sign objects. This creates the \*OBJECTSIGNING certificate store, which you use to manage object signing certificates.

**Note:** Although this scenario does not use object signing certificates, be sure to complete this step. If you cancel at this point in the task, the task ends and you must perform separate tasks to complete your SSL certificate configuration.

- h. Select the applications that you want to trust the local CA.

**Note:** Be sure to select the application ID for the i5/OS TCP/IP FTP server (QIBM\_QTMF\_FTP\_SERVER).

### Enable Secure Sockets Layer for MyCo's FTP server:

Now that the File Transfer Protocol (FTP) server has a certificate assigned to it, MyCo configures the FTP server to use SSL by following these steps:

1. In iSeries Navigator, expand **the iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. Right-click **FTP**.
3. Select **Properties**.
4. Select the **General** tab.
5. Choose the following option for SSL support: **Secure only**. Select this to allow only SSL sessions with the FTP server. Connections can be made to the non-secure FTP port, but the FTP client must negotiate an SSL session before the user is allowed to log in.

With this task complete, MyCo's FTP server can now use SSL to encrypt communication sessions and protect the privacy of the data transmitted during these sessions. However, to configure the FTP client to participate in an SSL session with the FTP server, MyCo must provide their client, TheirCo, with a copy of the local CA certificate. To do this, MyCo needs to export a copy of the local CA certificate to a file and make the file available to TheirCo. After TheirCo has this file, they can use DCM to import the Local CA certificate into the \*SYSTEM certificate store, and configure the i5/OS FTP client to use SSL.

### Export a copy of MyCo's local Certificate Authority certificate to a file:

MyCo must provide TheirCo with a copy of the local Certificate Authority (CA) certificate. TheirCo's client application must be configured to trust the CA certificate before it can participate in an Secure Socket Layer (SSL) session.

MyCo uses the following steps to export a copy of the local CA certificate to a file:

1. Start DCM.
2. Click **Select a Certificate Store**.
3. Select **\*SYSTEM** as the certificate store to open and click **Continue**.
4. When the Certificate Store and Password page displays, provide the password that was specified for the certificate store when it was created, and click **Continue**.
5. After the navigation frame refreshes, select **Manage Certificates**, and then select the **Export certificate** task.
6. Select **Certificate Authority (CA)** and click **Continue** to display a list of CA certificates.

7. Select the MyCo local CA certificate from the list and click **Export**.
8. Specify **File** as the export destination and click **Continue**.
9. Specify a fully qualified path and file name for the exported Local CA certificate and click **Continue** to export the certificate.
10. Click **OK** to exit the Export confirmation page.

Now you can transfer these files to the iSeries endpoint systems on which you intend to verify signatures that you created with the certificate. You can use e-mail or FTP to transfer the files since they do not need to be sent securely.

#### **Create a \*SYSTEM certificate store on TheirCo's server:**

To participate in an SSL session, TheirCo's i5/OS File Transfer Protocol (FTP) client must be able to recognize and accept the certificate that MyCo's FTP server presents to establish the Secure Socket Layer (SSL) session. To authenticate the server certificate, TheirCo's FTP client must have a copy of the Certificate Authority (CA) certificate in the \*SYSTEM certificate store. The \*SYSTEM certificate store contains a copy of most public CA certificates. However, when MyCo's FTP server uses a certificate from a local CA, the TheirCo's FTP client must obtain a copy of the local CA certificate and import it into the \*SYSTEM certificate store.

This scenario assumes that Digital Certificate Manager (DCM) has not been previously used to create or manage certificates. Consequently, TheirCo must first create the \*SYSTEM certificate store by following these steps:

1. Start DCM.
2. In the Digital Certificate Manager (DCM) navigation frame, select **Create New Certificate Store** and select **\*SYSTEM** as the certificate store to create and click **Continue**.
3. Select **No** to create a certificate as part of creating the \*SYSTEM certificate store and click **Continue**.
4. Specify a password for the new certificate store and click **Continue** to display a confirmation page.
5. Click **OK**.

#### **Import MyCo's local CA certificate into TheirCo's \*SYSTEM certificate store:**

TheirCo uses these steps to import the local Certificate Authority (CA) certificate into the \*SYSTEM certificate store and specify that it is a trusted source for certificates:

1. In the DCM navigation frame, click **Select a Certificate Store** and select **\*SYSTEM** as the certificate store to open.
2. When the Certificate Store and Password page displays, provide the password that was specified for the certificate store when it was created, and click **Continue**.
3. After the navigation frame refreshes, select **Manage Certificates** to display a list of tasks.
4. From the task list, select **Import certificate**.
5. Select **Certificate Authority (CA)** as the certificate type and click **Continue**.
6. Specify the fully qualified path and file name for the CA certificate file and click **Continue**. A message displays that either confirms that the import process succeeded or provide error information if the process failed.

#### **Specify MyCo's local CA as a trusted CA for TheirCo's FTP client:**

Before TheirCo can use the File Transfer Protocol (FTP) client to make secure connections to the MyCo FTP server, TheirCo must use Digital Certificate Manager (DCM) to specify which Certificate Authorities the client should trust. This means that TheirCo must specify that the local CA certificate that was imported previously is to be trusted.

TheirCo uses the following steps to specify that their FTP client should trust MyCo's local CA certificate:

1. Start DCM.
2. Click **Select a Certificate Store** and select \*SYSTEM as the certificate store to open.
3. When the Certificate Store and Password page displays, provide the password that was specified for the certificate store when it was created, and click **Continue**.
4. In the navigation frame, select **Manage Applications** to display a list of tasks.
5. From the task list, select **Define CA trust list**.
6. Select **Client** as the type of application for which you want to define the list and click **Continue**.
7. Select the i5/OS TCP/IP FTP Client application (QIBM\_QTMF\_FTP\_CLIENT) from the list and click **Continue** to display a list of CA certificates.
8. Select MyCo's local CA certificate that was imported previously and click **OK**. DCM displays a message to confirm the trust list selection.

With these steps complete, MyCo's FTP server can establish an SSL session with TheirCo's FTP client and server. Refer to "Secure the FTP client with Transport Layer Security or Secure Socket Layer" on page 20.

---

## Configure the File Transfer Protocol server

You can set up an iSeries File Transfer Protocol (FTP) server for graphical FTP clients, Web browsers, and Web tools.

The TCP/IP Connectivity Utilities licensed program comes with TCP/IP FTP servers configured. When you start TCP/IP, the FTP server starts simultaneously. Before you configure an FTP server on the Internet, you should review these safeguards to protect your data:

- Use a firewall between your iSeries server and the Internet.
- Use a non-production iSeries for your FTP server.
- Do not attach the FTP server to the rest of your company's Local Area Networks (LANs) or Wide Area Network (WANs).
- Use FTP exit programs to secure access to the FTP server.
- Test FTP exit programs once a month to ensure that they do not contain security loopholes.
- Do not allow anonymous FTP users to have read and write access to the same directory. This permits the anonymous user to be untraceable on the Internet.
- Log all access to your iSeries FTP server and review the logs daily or weekly for possible attacks.
- Verify that the correct exit programs are registered for the FTP server once a month.
- Review Secure FTP for information about securing your iSeries FTP server.

### Related reference

"Secure File Transfer Protocol" on page 14

You can protect your data by securing File Transfer Protocol (FTP) with Secure Sockets Layer (SSL), monitoring FTP users, and managing user access to FTP functions.

## File Transfer Protocol server in iSeries Navigator

You can use iSeries Navigator to configure and manage the iSeries File Transfer Protocol (FTP) server.

You can use iSeries Navigator to work with your FTP server configuration. To access the graphical user interface for FTP in iSeries Navigator, follow these steps:

1. In the iSeries Navigator, expand **your iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. In the right pane, right-click **FTP** and select **Properties**.
3. From here, you can change the properties for your FTP server. You can view the online help by clicking the help buttons. To obtain help for a specific field, click the question mark button, and then click that field.

### Related tasks

“Start and stop the File Transfer Protocol server” on page 24

You can start and stop the File Transfer Protocol (FTP) server by using iSeries Navigator.

## Configure FTP servers for graphical FTP clients and Web tools

You can configure an File Transfer Protocol (FTP) server on the iSeries to support graphical FTP clients, Web browsers, and other Web tools.

Most graphical FTP clients use UNIX<sup>®</sup> as their list format and path file as their file name format. Follow these instructions to set the FTP server properties to use the supported formats:

1. In the iSeries Navigator, expand **your iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. In the right pane, right-click **FTP** and select **Properties**.
3. On the **Properties** page, click the **Initial Formats** tab.
  - Enable **Path** as the File Naming Format.
  - Enable **UNIX list format** as the File List Format.

**Note:** You can control the LISTFMT and NAMEFMT settings for specific FTP sessions using an exit program for the TCPL0200 format or TCPL0300 format of the FTP Server Logon Exit Point.

You can also change the list format *after* an FTP session is in progress with options for the FTP server SITE (Send Information Used by a Server System) subcommand. These settings control the results returned by the LIST (File List) and NLST (Name List) FTP server subcommands.

### Related reference

“TCPL0200 exit point format” on page 118

“TCPL0300 exit point format” on page 120

“SITE (Send Information Used by a Server System)” on page 52

“LIST (File List)” on page 46

“NLST (Name List)” on page 47

## Use the LIST subcommand information in iSeries format

**Note:** The iSeries client supports both the iSeries format and the UNIX-style format.

Here is the original iSeries style format for the LIST subcommand (when LISTFMT=0):

```
owner size date time type name
```

A blank space separates each field.

This is a description of each field:

- owner** The 10-character string that represents the user profile which owns the subject. This string is left-aligned, and includes blanks. This field is blank for anonymous FTP sessions.
- size** The 10-character number that represents the size of the object. This number is right-aligned, and it includes blanks. This field is blank when an object has no size associated with it.
- date** The 8-character modification date in the format that is defined for the server job. It uses date separators that are defined for the server job. This modification date is left-aligned, and it includes blanks.
- time** The 8-character modification time that uses the time separator, which the server job defines.
- type** The 10-character i5/OS object type.
- name** The variable length name of the object that follows a CRLF (carriage return, line feed pair). This name can include blanks.



Here is an example of the original iSeries style format:

```
BAILEYSE 5263360 06/11/97 12:27:39 *FILE BPTFSAVF
```

#### Related reference

“LIST information in UNIX-style format”

“SITE (Send Information Used by a Server System)” on page 52

“LIST (File List)” on page 46

“NLST (Name List)” on page 47

## LIST information in UNIX-style format

**Note:** The iSeries client supports both the iSeries format and the UNIX-style format.

Here is the UNIX style format for the LIST subcommand (when LISTFMT=1):

```
mode links owner group size date time name
```

A blank space separates each field.

This is a description of each field in the UNIX style format:

**mode** You can use 10 characters. Each character has a specific meaning.

The first character	Meaning
d	The entry is a directory.
b	The entry is a block special file.
c	The entry is a character special file.
l	The entry is a symbolic link. Either the -N flag was specified, or the symbolic link did not point to an existing file.
p	The entry is a first-in, first-out (FIFO) special file.
s	The entry is a local socket.
-	The entry is an ordinary file.

The next nine characters divide into three sets of three characters each. The three characters in each set indicate, respectively, read, write, and execute permission of the file. With execute permission of a directory, you can search a directory for a specified file. Indicate permissions like this: The first set of three characters show the owner’s permission. The next set of three characters show the permission of the other users in the group. The last set of three characters shows the permission of anyone else with access to the file.

The first character	Function
r	read
w	write (edit)
x	execute (search)
-	corresponding permission not granted

**links** The number of links to the object. The minimum number of characters is 3. The maximum number of characters is 5. The characters are right justified, and they include blanks.

**owner** The owner of the object. The minimum number of characters is 8. The maximum number of characters is 10. The characters are left justified, and they include blanks. This field contains the user profile name of the object owner. However, for anonymous FTP sessions, this field contains the owner ID number.

**group** The owner of the object. The minimum number of characters is 8. The maximum number of characters is 10. The characters are left justified, and they include blanks. This field contains the user profile name of the group. However, if there is no group, this field contains the group ID number. The field also contains the group ID number for an anonymous FTP session.

**size** The size of the object. The minimum number of characters is 7. The maximum number of characters is 10. The characters are right-aligned, and they include blanks. When there is no size for the object, the default is zero.

**datetime**

The 12 character modification time. The characters are left-aligned, and they include blanks. This is the format of this field when the modification time is within the previous 180 days:

Mmm dd hh:mm

This is the format of this field when the modification time is not within the previous 180 days:

Mmm dd yyyy

Here is the description of each field.

Characters	Meaning
Mmm	Abbreviated month.
dd	Two character day of the month. The characters are right justified and padded with blanks.
hh	Two-digit hour (00-23). The digits are right justified and padded with zeros.
mm	Two-digit minute (00-59). The digits are right justified and padded with zeros.
yyyy	Four-digit year.

**name** The variable length name of the object, which precedes a CRLF (carriage return, line feed pair). The name may include blanks.

Here is an example of the UNIX style format:

```
drwxrwxrwx 4 QSYS 0 51200 Feb 9 21:28 home
```

Consider this information as you work with UNIX format data that is returned by the LIST subcommand: When LISTFMT=1, the LIST content varies for QSYS.LIB files depending on the NAMEFMT setting:

- When NAMEFMT=1, you will see only the QSYS.LIB file names.
- When NAMEFMT=0, you will see both the QSYS.LIB file names and the names of the members in the file or files.

**Related reference**

“Use the LIST subcommand information in iSeries format” on page 8

“SITE (Send Information Used by a Server System)” on page 52

“LIST (File List)” on page 46

“NLST (Name List)” on page 47

## Configure anonymous File Transfer Protocol

Anonymous File Transfer Protocol (FTP) enables remote users to use the FTP server without an assigned user ID and password.

Anonymous FTP enables unprotected access (no password required) to selected information about a remote system. The remote site determines what information is made available for general access. Such



information is considered to be publicly accessible and can be read by anyone. It is the responsibility of the person who owns the information and the system to assure that only appropriate information is made available.

To access this information, a user logs on to the hosts using the user ID ANONYMOUS. The user ANONYMOUS has limited access rights to the files on the server and has some operating restrictions. Typically, the following operations are only operations allowed.

- Logging on using FTP
- Listing the contents of a limited set of directories
- Retrieving files from these directories.

Typically, anonymous users are not allowed to transfer files to the FTP server system. Some systems do provide an incoming directory for anonymous users to send data to. Traditionally, the special anonymous user account accepts a string as a password, although it is common to use either the password 'guest' or one's e-mail address. Some archive sites explicitly ask for the user's e-mail address and do not allow logon with the guest password. Providing an e-mail address is a courtesy that allows the archive site operators to get some idea of who is using their services.

## **Anonymous FTP on the iSeries**

The File Transfer Protocol (FTP) server does not use anonymous FTP. To set up anonymous FTP on the iSeries server, you need to provide exit programs for the FTP server logon exit point and the request validation exit point.

You might want to provide anonymous FTP because it is a convenient, and often necessary service. However, using anonymous FTP raises security concerns for the iSeries server.

### **Related concepts**

"Manage access using File Transfer Protocol exit programs" on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

"Control File Transfer Protocol access" on page 16

If you are using File Transfer Protocol (FTP), you need to control users to protect your data and network. This topic offers tips and security considerations.

### **Related reference**

"Server logon exit point" on page 100

You can control the authentication of users to a TCP/IP application server with the TCP/IP Application Server Logon exit point.

## **Prepare for anonymous File Transfer Protocol**

You might need to know how to set up your anonymous File Transfer Protocol (FTP) and ensure the security of your FTP.

## **Skill requirements**

To set up anonymous FTP, you need the following skills:

- Familiarity with the iSeries character-based interface and commands with multiple parameters and keywords.
- Ability to create libraries, members, and source physical files on your iSeries (you should have at least \*SECOFR authority).
- Ability to assign authorities to libraries, files, members, and programs.
- Ability to write, change, compile, and test programs on your iSeries server.

## Security considerations

The first step in implementing anonymous FTP is to define your anonymous FTP server site policy. This plan defines the FTP site security and determines how to code your exit programs. Because the FTP server will allow anyone to access your data, you must carefully consider how you want it to be used, and what data must be protected.

Review the following guidelines for your FTP site policy plan:

- Use a firewall between your iSeries server and the Internet.
- Use a non-production iSeries for your FTP server.
- Do not attach the FTP server to the rest of your company's LANs or WANs.
- Use FTP exit programs to secure access to the FTP server.
- Test FTP exit programs to ensure that they do not contain security loopholes.
- Do not allow anonymous FTP users to have read and write access to the same directory. This permits the anonymous user to be untraceable on the Internet.
- Allow ANONYMOUS access only. Do not allow any other userids and do not authenticate passwords.
- Restrict ANONYMOUS access to one public library or directory only. (Where will it be? What will you call it?)
- Place only public access files in the public library or directory.
- Restrict ANONYMOUS users to 'view' and 'retrieve' subcommands only (get, mget). **Do not under any circumstances allow ANONYMOUS users to use CL commands.**
- Log all access to your iSeries FTP server.
- Review FTP server logs daily or weekly for possible attacks.
- Verify that the FTP server registers the correct exit programs once a month.
- Test the FTP server for security holes once a month.

## Write exit programs for anonymous File Transfer Protocol

To use anonymous File Transfer Protocol (FTP), you need to write two exit programs: FTP Server Logon exit program and FTP Server Request Validation exit program.

The FTP Server Logon exit program enables the ANONYMOUS user ID and forces the ANONYMOUS user to the public library or directory. The FTP Server Request Validation exit program restricts the commands, files, and directories or libraries that the ANONYMOUS user can use.

## Exit points and exit point formats

The FTP server communicates with each exit program through a specific exit point. Parameters are passed between the server and the exit program. The format of the exchanged information is specified by an exit point format.

Program	Exit Point	Format
Server logon	QIBM_QTMF_SVR_LOGON	TCPL0100, TCPL0200, or TCPL0300. <sup>1</sup>
Request validation	QIBM_QTMF_SERVER_REQ	VLRQ0100

<sup>1</sup> An exit point might have more than one format, but an exit program can only be registered for one of the exit point formats. Examine each of these formats, then choose the one most appropriate for your system.

## Example programs

Example programs are available to help you set up anonymous FTP on your server. You can use these samples as a starting point to build your own programs. By copying portions of the code from the

samples, you can add them to programs that you write yourself. It is recommended that you run the sample programs on a system other than your production system.

**Note:** These examples are for illustration purposes only. They do not contain enough features to run on a production machine as is. Feel free to use them as a starting point, or to use sections of code as you write your own programs.

#### **Related concepts**

“Request validation exit point: Client and server” on page 88

The Request Validation exit points can be used to restrict operations which can be performed by FTP users. Request validation exit points are provided by both the FTP client and server; to restrict both FTP client and FTP server access, exit programs must be added to both exit points.

#### **Related reference**

“File Transfer Protocol exit programs” on page 87

You can use File Transfer Protocol (FTP) exit programs to secure FTP. The FTP server communicates with each exit program through a specific exit point. This topic includes parameter descriptions and code examples.

“Server logon exit point” on page 100

You can control the authentication of users to a TCP/IP application server with the TCP/IP Application Server Logon exit point.

### **Create an i5/OS user profile: ANONYMOUS**

To prevent anyone from directly signing on to the iSeries server with the user profile ANONYMOUS, you need to create a user profile of ANONYMOUS and assign it a password of \*NONE.

You can create this profile using iSeries Navigator.

1. In **iSeries Navigator**, expand **Users and Groups**.
2. Right-click **All Users** and select **New Users**.
3. On the New Users panel, enter the following information:  
**User name** = ANONYMOUS and  
**Password** = No password.
4. Click the **Jobs** button and select the General tab.
5. On the General tab, assign the current library and home directory that the anonymous user should use.
6. Click **OK** and complete any other settings.
7. Click **Add** to create the profile.

#### **Related concepts**

iSeries Navigator

### **Create a public library or directory**

After creating anonymous users, you might want to create a public library or directory for them to use. Typically anonymous users should only be able to access public files.

It is recommended that you restrict anonymous users to a single library or a single directory tree, which only contain “public” files.

1. Create the public libraries or directories that will contain files accessible through anonymous File Transfer Protocol (FTP).
2. Load your public libraries or directories with the public access files.
3. Set the public libraries or directories and file authorities to PUBLIC \*USE.

## Install and register exit programs

You can create a library to contain your exit programs and their log files, compile the programs, and register them for use by the File Transfer Protocol (FTP) server.

### Related concepts

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

### Related tasks

“Remove exit programs” on page 123

You can remove an installed exit program.

### Related reference

“File Transfer Protocol exit programs” on page 87

You can use File Transfer Protocol (FTP) exit programs to secure FTP. The FTP server communicates with each exit program through a specific exit point. This topic includes parameter descriptions and code examples.

### Install the exit program:

1. Create a library to contain your exit programs and their log files.
2. Compile your exit programs in this library.
3. Grant PUBLIC \*EXCLUDE authority to the library, program, and file objects.

The FTP server application adopts authority when necessary to resolve and call the exit program.

### Register the exit program:

1. At the iSeries character-based interface, enter WRKREGINF.
2. Page down to an FTP Server Logon exit point:

```
QIBM_QTMF_SVR_LOGON  TCPL0100
QIBM_QTMF_SVR_LOGON  TCPL0200
QIBM_QTMF_SVR_LOGON  TCPL0300
QIBM_QTMF_SERVER_REQ VLRQ0100
```

3. Enter 8 in the Opt field to the left of the exit point entry and press Enter.
4. At the Work with Exit Programs display, enter a 1(add).
5. Enter the name of the exit program in the Exit Program field.
6. Enter the name of the library that contains the exit program in the Library field.
7. Press Enter.
8. End and restart the FTP server to ensure that all FTP server instances use the exit programs.
9. Test your exit programs thoroughly.

**Note:** Exit programs take effect as soon as the FTP server requests a new FTP session. Sessions that are already running are not affected.

---

## Secure File Transfer Protocol

You can protect your data by securing File Transfer Protocol (FTP) with Secure Sockets Layer (SSL), monitoring FTP users, and managing user access to FTP functions.

If you use your iSeries system as an FTP server on the Internet, it is accessible to the entire world. Therefore, attention to FTP security is necessary to ensure that vital business data stored on your iSeries server is not compromised. There are also steps you can take to protect your FTP client.

### Related concepts

“Configure the File Transfer Protocol server” on page 7

You can set up an iSeries File Transfer Protocol (FTP) server for graphical FTP clients, Web browsers, and Web tools.

## Prevent File Transfer Protocol server access

You can block the File Transfer Protocol (FTP) port by reading this topic.

If you do not want anyone to use FTP to access your iSeries server, you should prevent the FTP server from running. To prevent FTP access to your iSeries, follow these steps:

### Prevent the File Transfer Protocol server from starting automatically

To prevent File Transfer Protocol (FTP) server jobs from starting automatically when you start TCP/IP, follow these steps:

1. In iSeries Navigator, expand **your iSeries Server** → **Network** → **Servers** → **TCP/IP** .
2. Right-click **FTP** and select **Properties**.
3. Deselect **Start when TCP/IP starts**.

### Prevent access to File Transfer Protocol ports

To prevent File Transfer Protocol (FTP) from starting, and to prevent someone from associating a user application, such as a socket application, with the port that the iSeries normally uses for FTP, follow these steps:

1. In iSeries Navigator, expand **your iSeries Server** → **Network** → **Servers** → **TCP/IP** .
2. Right-click **TCP/IP Configuration** and select **Properties**.
3. In the **TCP/IP Configuration Properties** window, click the **Port Restrictions** tab.
4. On the **Port Restrictions** page, click **Add**.
5. On the **Add Port Restriction** page, specify the following information:
  - **User name:** Specify a user profile name that is protected on your iSeries. (A protected user profile is a user profile that does not own programs that adopt authority and does not have a password that is known by other users.) By restricting the port to a specific user, you automatically exclude all other users.
  - **Starting port:** 20
  - **Ending port:** 21
  - **Protocol:** TCP
6. Click **OK** to add the restriction.
7. On the **Port Restrictions** page, click **Add** and repeat the procedure for the UDP protocol.
8. Click **OK** to save your port restrictions and close the **TCP/IP Configuration Properties** window.
9. The port restriction takes effect the next time that you start TCP/IP. If TCP/IP is active when you set the port restrictions, you should end TCP/IP and start it again.

#### Notes:

- The port restriction takes effect the next time that you start TCP/IP. If TCP/IP is active when you set the port restrictions, you should end TCP/IP and start it again.
- The Internet Assigned Numbers Authority (IANA) website provides information about assigned port numbers at <http://www.iana.org>.
- If ports 20 or 21 are restricted to a user profile other than QTCP, attempting to start the FTP server will cause it to immediately end with errors.
- This method works only for completely restricting an application such as the FTP server. It does not work for restricting specific users. When a user connects to the FTP server, the request uses the QTCP profile initially. The system changes to the individual user profile after the connection is successful. Every user of the FTP server uses QTCP's authority to the port.

## Control File Transfer Protocol access

If you are using File Transfer Protocol (FTP), you need to control users to protect your data and network. This topic offers tips and security considerations.

If you want to allow FTP clients to access your system, be aware of the following security concerns:

- Your object authority scheme might not provide detailed enough protection when you allow FTP on your system. For example, when a user has the authority to view a file (\*USE authority), the user can also copy the file to a PC or to another system. You might want to protect some files from being copied to another system.
- You can use FTP exit programs to restrict the FTP operations that users can perform. You can use the FTP Request Validation Exit to control what operations you allow. For example, you can reject GET requests for specific database files.
- You can use the Server logon exit point to authenticate users who log on to the FTP server. Configure anonymous FTP describes how to use exit programs to set up support for anonymous FTP on your system.
- Unless you use Transport Layer Security (TLS) or Secure Socket Layer (SSL), FTP passwords are not encrypted when they are sent between the client system and the server system. Depending on your connection methods, your system might be vulnerable to password theft through line sniffing.
- If the QMAXSGNACN system value is set to 1, the QMAXSIGN system value applies to TELNET but not to FTP. If QMAXSGNACN is set to 2 or 3 (values which disable the profile if the maximum sign on count is reached), FTP logon attempts are counted. In this case, a hacker can mount a denial of service attack through FTP by repeatedly attempting to log on with an incorrect password until the user profile is disabled.
- For each unsuccessful attempt, the system writes message CPF2234 to the QHST log. You can write a program to monitor the QHST log for the message. If the program detects repeated attempts, it can end the FTP servers.
- You can use the Inactivity timeout (INACTTIMO) parameter on the FTP configuration to reduce the exposure when a user leaves an FTP session unattended. Be sure to read the documentation or online help to understand how the INACTTIMO parameter and the connection timer (for server startup) work together.

**Note:** The QINACTITV system value does not affect FTP sessions.

- When you use FTP batch support, the program must send both the user ID and the password to the server system. Either the user ID and password must be coded in the program, or the program must retrieve them from a file. Both these options for storing passwords and user IDs represent a potential security exposure. If you use FTP batch, you must ensure that you use object security to protect the user ID and password information. You should also use a single user ID that has limited authority on the target system. It should have only enough authority to perform the function that you want, such as file transfer.
- FTP provides remote-command capability, just as advanced program-to-program communications (APPC) and iSeries Access for Windows<sup>®</sup> do. The RCMD (Remote Command) FTP-server subcommand is the equivalent of having a command line on the system. Before you allow FTP, you must ensure that your object security scheme is adequate. You can also use the FTP exit program to limit or reject attempts to use the RCMD subcommand. FTP exit programs describes this exit point and provides sample programs.
- A user can access objects in the integrated file system with FTP. Therefore, you need to ensure that your authority scheme for the integrated file system is adequate when you run the FTP server on your system.
- A popular hacker activity is to set up an unsuspecting site as a repository for information. Sometimes, the information might be illegal or pornographic. If a hacker gains access to your site through FTP, the hacker uploads this undesirable information to your iSeries. The hacker then informs other hackers of your FTP address. They in turn access your iSeries with FTP and download the undesirable information.



You can use the FTP exit programs to help protect against this type of attack. For example, you might direct all requests to upload information to a directory that is write-only. This defeats the hacker's objective because the hacker's friends will not be able to download the information in the directory.

#### **Related concepts**

"Configure anonymous File Transfer Protocol" on page 10

Anonymous File Transfer Protocol (FTP) enables remote users to use the FTP server without an assigned user ID and password.

#### **Related reference**

"Server logon exit point" on page 100

You can control the authentication of users to a TCP/IP application server with the TCP/IP Application Server Logon exit point.

"Run File Transfer Protocol in unattended mode using a batch job" on page 30

This topic provides examples of how to run File Transfer Protocol (FTP) in an unattended mode.

"File Transfer Protocol exit programs" on page 87

You can use File Transfer Protocol (FTP) exit programs to secure FTP. The FTP server communicates with each exit program through a specific exit point. This topic includes parameter descriptions and code examples.

#### **Related information**

AS/400 Internet Security: Protecting Your AS/400 from HARM in the Internet

## **Use Secure Sockets Layer to secure the File Transfer Protocol server**

With Secure Sockets Layer (SSL) you can eliminate the exposure of transmitting passwords and data in the clear when using the i5/OS File Transfer Protocol (FTP) server with an FTP client that also uses SSL.

The FTP server provides enhanced security while sending and receiving files over a untrusted network. FTP server uses SSL to secure passwords and other sensitive data during an information exchange. The FTP server supports either SSL or TLS protected sessions, including client authentication and automatic sign-on.

Most SSL-enabled applications connect a client to separate TCP ports, one port for "unprotected" sessions and the other for secure sessions. However, secure FTP is a bit more flexible. A client can connect to a non-encrypted TCP port (typically TCP port 21), and then negotiate authentication and encryption options. A client can also choose a secure FTP port (typically TCP port 990), where connections are assumed to be SSL. The iSeries FTP server provides for both of these options.

Before you can configure the FTP server to use SSL, you must have installed the prerequisite programs and set up digital certificates on your iSeries.

**Note:** Create a local Certificate Authority or use DCM to configure the FTP server to use a public certificate for SSL.

#### **Related concepts**

Secure Sockets Layer (SSL)

SSL concepts

Prerequisite programs

"Secure the FTP client with Transport Layer Security or Secure Socket Layer" on page 20

You can use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) connections to encrypt data transferred over File Transfer Protocol (FTP) control and data connections.

#### **Related tasks**

Set up digital certificates

Use a public certificate

## Create a local Certificate Authority

You can use the IBM Digital Certificate Manager (DCM) to create and operate a local Certificate Authority (CA) on your iSeries server. A local CA enables you to issue private certificates for applications that run on your iSeries server.

To use DCM to create and operate a local CA on the iSeries server, follow these steps:

1. Start DCM.
2. In the navigation frame of DCM, select **Create a Certificate Authority (CA)** to display a series of forms. These forms guide you through the process of creating a local CA and completing other tasks needed to begin using digital certificates for SSL, object signing, and signature verification.
3. Complete all the forms that display. There is a form for each of the tasks that you need to perform to create and operate a local CA on the iSeries server. Completing these forms allows you to:
  - a. Choose how to store the private key for the local CA certificate. This step is included only if you have an IBM 4758-023 PCI Cryptographic Coprocessor installed on your iSeries. If your system does not have a cryptographic coprocessor, DCM automatically stores the certificate and its private key in the local CA certificate store.
  - b. Provide identifying information for the local CA.
  - c. Install the local CA certificate on your PC or in your browser. This enables software to recognize the local CA and validate certificates that the CA issues.
  - d. Choose the policy data for your local CA.
  - e. Use the new local CA to issue a server or client certificate that applications can use for SSL connections. If you have an IBM 4758-023 PCI Cryptographic Coprocessor installed in the iSeries server, this step allows you to select how to store the private key for the server or client certificate. If your system does not have a coprocessor, DCM automatically places the certificate and its private key in the \*SYSTEM certificate store. DCM creates the \*SYSTEM certificate store as part of this task.
  - f. Select the applications that can use the server or client certificate for SSL connections.

**Note:** Be sure to select the application ID for the i5/OS FTP Server (QIBM\_QTMF\_FTP\_SERVER).

- g. Use the new local CA to issue an object signing certificate that applications can use to digitally sign objects. This creates the \*OBJECTSIGNING certificate store, which you use to manage object signing certificates.

**Note:** Although this scenario does not use object signing certificates, be sure to complete this step. If you cancel at this point in the task, the task ends and you must perform separate tasks to complete your SSL certificate configuration.

- h. Select the applications that you want to trust the local CA.

**Note:** Be sure to select the application ID for the i5/OS FTP Server (QIBM\_QTMF\_FTP\_SERVER).

### Related tasks

Start DCM

Manage user certificates

Use APIs to programmatically issue certificates to non-iSeries users

Obtain a copy of the private CA certificate

## Associate a certificate with the File Transfer Protocol server

Perform this task if you did not assign a certificate to the File Transfer Protocol (FTP) server application during the creation of the local Certificate Authority (CA), or if you have configured your system to request a certificate from a public CA.

1. Start IBM Digital Certificate Manager. If you need to obtain or create certificates, or otherwise setup or change your certificate system, do so now. See Configure DCM for information about setting up a certificate system.



2. Click the **Select a Certificate Store** button.
3. Select **\*SYSTEM**. Click **Continue**.
4. Enter the appropriate password for \*SYSTEM certificate store. Click **Continue**.
5. When the left navigational menu reloads, expand **Manage Applications**.
6. Click **Update certificate assignment**.
7. On the next screen, select **Server** application. Click **Continue**.
8. Click **i5/OS TCP/IP FTP Server**.
9. Click **Update Certificate Assignment** to assign a certificate to this FTP Server.
10. Select a certificate from the list to assign to the server.
11. Click **Assign New Certificate**.
12. DCM reloads to the **Update Certificate Assignment** page with a confirmation message. When you are finished setting up the certificates for the FTP server, click **Done**.

#### **Related tasks**

Start DCM

“Enable Secure Socket Layer on the File Transfer Protocol server” on page 20

In order to use Secure Socket Layer (SSL) to secure your File Transfer Protocol (FTP) server, you need to complete the configuration steps first.

### **Require client authentication for the File Transfer Protocol server**

If you need the File Transfer Protocol (FTP) server to authenticate clients, you can change the application specifications in IBM Digital Certificate Manager (DCM). This step is optional.

**Note:** With the FTP server you can authenticate clients, but you cannot do so with the i5/OS FTP client. You can require client authentication, but it will exclude connections that are for i5/OS FTP clients.

If an FTP client connects and client authentication is enabled for the server, the client must still send a USER subcommand. After the USER subcommand information is sent, the FTP server will check that the user matches the profile associated with the client certificate that the client sent to the server as part of the SSL handshake. If the user matches the client certificate, no password is needed and the FTP server will log the user onto the system. The USER subcommand is needed because there is no mechanism in the FTP protocol to “inform” the client that it’s logged on without the command.

1. Start IBM Digital Certificate Manager. If you need to obtain or create certificates, or otherwise setup or change your certificate system, do so now. See Configure DCM for information about setting up a certificate system.
2. Click the **Select a Certificate Store** button.
3. Select **\*SYSTEM**. Click **Continue**.
4. Enter the appropriate password for \*SYSTEM certificate store. Click **Continue**.
5. When the left navigational menu reloads, expand **Manage Applications**.
6. Click **Update application definition**.
7. On the next screen, select **Server** application. Click **Continue**.
8. Click **i5/OS TCP/IP FTP Server**.
9. Click **Update Application Definition**.
10. In the table that displays, select **Yes** to require client authentication.
11. Click **Apply**.
12. DCM reloads to the **Update Application Definition** page with a confirmation message. When you are finished updating the application definition for the FTP server, click **Done**.

#### **Related tasks**

Start DCM

## Enable Secure Socket Layer on the File Transfer Protocol server

In order to use Secure Socket Layer (SSL) to secure your File Transfer Protocol (FTP) server, you need to complete the configuration steps first.

Perform the following steps to enable SSL on the FTP server:

1. In iSeries Navigator, expand **your iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. Right-click **FTP**.
3. Select **Properties**.
4. Select the **General** tab.
5. Choose one of these options for SSL support:
  - **Secure only**  
Select this to allow only SSL sessions with the FTP server. Connections can be made to the non-secure FTP port, but the FTP client must negotiate an SSL session before the user is allowed to log in.
  - **Non-secure only**  
Select this to prohibit secure sessions with the FTP server. Attempts to connect to an SSL port will not connect.
  - **Both secure and non-secure**  
Allows both secure and non-secure sessions with the FTP server.

**Note:** You do not need to restart the FTP server. It will dynamically detect that a certificate has been assigned to it. If it does not dynamically detect this change, verify that you have the latest PTFs applied to your iSeries server.

### Related tasks

“Associate a certificate with the File Transfer Protocol server” on page 18

Perform this task if you did not assign a certificate to the File Transfer Protocol (FTP) server application during the creation of the local Certificate Authority (CA), or if you have configured your system to request a certificate from a public CA.

## Secure the FTP client with Transport Layer Security or Secure Socket Layer

You can use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) connections to encrypt data transferred over File Transfer Protocol (FTP) control and data connections.

The primary reason for encryption on the control connection is to conceal the password when logging on to the FTP server.

Before using the FTP client to make secure connections to servers, you must use DCM to configure trusted certificate authorities for the FTP Client. Any certificate authorities which were used to create certificates assigned to servers that you want to connect to must be added. Exporting or importing Certificate Authority (CA) certificates might be required depending on the CAs used. Refer to Define a CA trust list for an application in the DCM topic for more information about CA trusted authorities.

If you choose TLS or SSL encryption for the control connection, the FTP client will also encrypt the data sent on the FTP data connection by default. FTP protocol does not allow you to have a secure data connection without a secure control connection.

Encryption can have a significant performance cost and can be bypassed on the data connection. This allows you to transfer non-sensitive files without decreasing performance and still protect the system's security by not exposing passwords.

The FTP client has parameters for the STRTCPFTP CL command and subcommands which are used as part of the TLS or SSL support (SECOpen and SECData).

## Specifying Transport Layer Security or Secure Socket Layer protection for the iSeries FTP client

### Control connection

TLS/SSL protection can be specified on the STRTCPFTP command and the SECOPEN subcommand.

For the STRTCPFTP (FTP) command, specify \*SSL for the SECCNN secure connection parameter to request a secure control connection. Also, you might be able to specify \*IMPLICIT to obtain a secure connection on a pre-defined server port number.

Within your FTP client session, the SECOPEN subcommand can be used to obtain a secure control connection.

### Data connection

For the STRTCPFTP (FTP) command, enter \*PRIVATE for the DTAPROT data protection parameter to specify a secure data connection. Enter \*CLEAR for the DTAPROT data protection parameter to specify data to be sent without encryption.

When you have a secure control connection, you can use the SECADATA subcommand to change the data connection protection level.

### Implicit SSL connection

Some FTP servers support what is called an implicit SSL connection. This connection provides the same encryption protection as the \*SSL option, but can only be done on a pre-determined server port, typically 990, for which the server must be configured to expect an SSL or TLS connection negotiation.

This method is provided to allow secure connections to those FTP implementations that cannot support the standard protocol for providing TLS or SSL protection.

Many early implementations of SSL support used the implicit approach, but now it has been deprecated by the IETF.

### Note:

The standard protocol for setting up an TLS or SSL connection requires that the AUTH (Authorization) server subcommand be used when making the connection to the server. Also, the server subcommands PBSZ and PROT are used to specify the data protection level.

However, for an implicit SSL connection, the AUTH, PBSZ, and PROT server subcommands are **not** sent to the server. Instead, the server will act as if the client had sent these subcommands with the parameters shown below:

- AUTH SSL
- PBSZ 0
- PROT P

### Related concepts

“Use Secure Sockets Layer to secure the File Transfer Protocol server” on page 17

With Secure Sockets Layer (SSL) you can eliminate the exposure of transmitting passwords and data in the clear when using the i5/OS File Transfer Protocol (FTP) server with an FTP client that also uses SSL.

### Related tasks

Define a CA trust list for an application

### Related reference

“Start and stop a client session” on page 26

You can start and stop a client session from this topic.

“SECOpen (Setting Data Security Protection)” on page 80

“SECDData (Setting Data Security Protection)” on page 79

#### **Related information**

“Scenario: Secure File Transfer Protocol with Secure Sockets Layer” on page 3

Use Secure Sockets Layer (SSL) to secure data being transferred to your partner company.

## **Manage access using File Transfer Protocol exit programs**

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

FTP provides a security level based on the i5/OS object security. This means that remote users cannot log on to the iSeries FTP server unless they have a valid user profile and password.

You can provide additional security by adding FTP exit programs to the FTP Server and Client exit points to further restrict FTP access to your system. For example, you can restrict FTP logon capability, as well as access to libraries, objects, and the use of commands.

You can write an FTP Server Request Validation exit program to restrict the CL commands and FTP subcommands that users can access.

You can control the authentication of users to a TCP/IP application server with an exit program for the Server logon exit point.

You can write an FTP Client Request Validation exit program for the Client exit point: Request validation. This controls which FTP client functions a user may perform.

Depending on your situation, you might consider limiting access to FTP subcommands using Application Administration Limit Access as an alternative to writing exit programs for the FTP Server Request Validation and FTP Client Request Validation exit points.

To allow the exit programs to work properly, you must Install and register your exit point programs. If your programs are no longer needed, you must properly Remove the exit point programs to prevent their future functioning.

#### **Related concepts**

“Configure anonymous File Transfer Protocol” on page 10

Anonymous File Transfer Protocol (FTP) enables remote users to use the FTP server without an assigned user ID and password.

“Request validation exit point: Client and server” on page 88

The Request Validation exit points can be used to restrict operations which can be performed by FTP users. Request validation exit points are provided by both the FTP client and server; to restrict both FTP client and FTP server access, exit programs must be added to both exit points.

#### **Related tasks**

“Manage access using iSeries Navigator” on page 23

You can limit access to File Transfer Protocol (FTP) server or client by using Application Administration in iSeries Navigator. Application Administration is component of iSeries Navigator that you can choose to install.

“Install and register exit programs” on page 14

You can create a library to contain your exit programs and their log files, compile the programs, and register them for use by the File Transfer Protocol (FTP) server.

“Remove exit programs” on page 123

You can remove an installed exit program.

### Related reference

“File Transfer Protocol exit programs” on page 87

You can use File Transfer Protocol (FTP) exit programs to secure FTP. The FTP server communicates with each exit program through a specific exit point. This topic includes parameter descriptions and code examples.

“Server logon exit point” on page 100

You can control the authentication of users to a TCP/IP application server with the TCP/IP Application Server Logon exit point.

“VLRQ0100 exit point format” on page 97

## Manage access using iSeries Navigator

You can limit access to File Transfer Protocol (FTP) server or client by using Application Administration in iSeries Navigator. Application Administration is component of iSeries Navigator that you can choose to install.

You can use iSeries Navigator to limit user access to FTP server and client functions. Use Application Administration to grant and deny access to functions for individual users or for groups of users. Alternatively, you can manage access to FTP functions by writing FTP exit programs for the FTP Request Validation Exit Points.

To manage user access to functions using iSeries Navigator, complete the following steps:

1. In iSeries Navigator, right-click **your iSeries server** and select **Application Administration**.
2. Select the **Host Applications** tab.
3. Expand **TCP/IP Utilities for iSeries**.
4. Expand **File Transfer Protocol (FTP)**.
5. Expand **FTP Client** or **FTP Server**.
6. Select the function that you want to allow or deny access to.
7. Click **Customize**.
8. Use the **Customize Usage** dialog to change the list of users and groups that are allowed or denied access to the function.
9. Click **OK** to save changes to the **Customize Access** page.
10. Click **OK** to exit the **Application Administration** page.

Alternatively, you can manage the access that a specific user or group has to registered FTP functions through iSeries Navigator’s Users and Groups management tool. To do this, follow these steps:

1. In iSeries Navigator, expand **your iSeries server** → **Users and Groups**.
2. Select **All Users** or **Groups**.
3. Right-click a user or group, then select **Properties**.
4. Click **Capabilities**.
5. Click **Applications**.

From here, you can change the user or group’s settings for the listed function. You can also change the settings for all functions in a hierarchy grouping by changing the settings of the “parent” function.

### Related concepts

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

## Monitor incoming File Transfer Protocol users

You can monitor who is logging in to your File Transfer Protocol (FTP) server.

Logging and reviewing FTP use allow you to monitor activity and check for outside attacks. To monitor for incoming FTP users, follow these steps:

1. In iSeries Navigator, expand **your server** → **Network** → **Servers** → **TCP/IP**.
2. In the right pane, right-click **FTP** and select **Server Jobs**.
3. The FTP server job panel appears. The Current user column displays the user logged in to the server job. If no user is logged in, Qtcp is displayed. Press F5 or select **View** → **Refresh** to update the display.

The format for the names of these jobs is *QTFTPnnnnnn*. The *nnnnnn* is a randomly-generated number.

To start FTP server jobs, follow the steps in Start the FTP server.

#### **Related tasks**

“Start and stop the File Transfer Protocol server”

You can start and stop the File Transfer Protocol (FTP) server by using iSeries Navigator.

---

## **Manage the File Transfer Protocol server**

You can manage the File Transfer Protocol (FTP) server, to start and stop the server, administer FTP security, and to use Secure Socket Layer (SSL).

You can set up your iSeries server to send, receive, and share files across networks by using file transfer protocol (FTP). FTP consists of two parts: the FTP client and the FTP server. You interact with the FTP client. The FTP client interacts with the FTP server. You do not normally interact directly with the FTP server. The following topics can help you to administer your FTP server.

## **Start and stop the File Transfer Protocol server**

You can start and stop the File Transfer Protocol (FTP) server by using iSeries Navigator.

For instructions on how to access FTP, see “File Transfer Protocol server in iSeries Navigator” on page 7.

#### **Related concepts**

“Determine problems with File Transfer Protocol” on page 143

You can view a list of steps to determine whether your Simple Mail Transfer Protocol (SMTP) is working correctly.

#### **Related tasks**

“Monitor incoming File Transfer Protocol users” on page 23

You can monitor who is logging in to your File Transfer Protocol (FTP) server.

“File Transfer Protocol server in iSeries Navigator” on page 7

You can use iSeries Navigator to configure and manage the iSeries File Transfer Protocol (FTP) server.

“Remove exit programs” on page 123

You can remove an installed exit program.

## **Start the File Transfer Protocol server**

To start the File Transfer Protocol (FTP) server, complete the following steps:

1. In iSeries Navigator, expand **your iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. In the right pane, right-click **FTP** and select **Start**.

## **Stop the File Transfer Protocol server**

To stop the File Transfer Protocol (FTP) server, complete the following steps:

1. In iSeries Navigator, expand **your iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. In the right pane, right-click **FTP** and select **Stop**.



## Set number of available File Transfer Protocol servers

You can specify the minimum number of available servers to be kept ready for future client connections.

To set this value, go to the **FTP Properties** page and specify a number from 1 to 20 for the **Initial number of servers to start**. Specifying a value of 1 delays incoming connections to the File Transfer Protocol (FTP) server. The recommended value is 3.

When a client connects to an iSeries FTP server, the server examines the number of active servers that are not connected to a client and the value specified for the initial number of servers to start. If the initial server value is greater than the number of available servers, additional servers are started so that the two numbers are equal. If the initial server value is less than the number of available servers, no action is taken. Changes to the initial server value take effect at the time of the next client connection, when the above process is activated.

For example, if there are five FTP client sessions established at the same time and the initial server value is set at 10, there will be 15 FTP servers running. The 15 servers include five servers for the five active client sessions and ten available servers. The number of available servers can be larger than the initial server value. In this same example, if the five clients end their sessions and no other sessions are started, there will be 15 available servers.

## Improve FTP server performance with configurable subsystem support

The default subsystem (QSYS/QSYSWRK) is used for many IBM-supplied server jobs.

Using a different subsystem than the default subsystem can result in improved File Transfer Protocol (FTP) performance because the need to share resources is eliminated.

To configure a subsystem for the FTP server, follow these steps:

1. In iSeries Navigator, expand **your iSeries Server** → **Network** → **Servers** → **TCP/IP**.
2. Right-click **FTP** and select **Properties**.
3. On the **FTP Properties** page, select **Subsystem description**.
4. Specify a subsystem description and a predefined library.

If the specified subsystem does not exist, then FTP will create it along with routing table entries and job descriptions. When the startup job for the server is started, it will specify the parameters for the newly created subsystem and then submit the server jobs for batch startup in that subsystem.

---

## Use the File Transfer Protocol client on iSeries server

You can start and stop client sessions, transfer and receive files, as well as set up File Transfer Protocol (FTP) batch jobs.

The FTP client allows you to transfer files that are found on your iSeries server, including those in the Root, QSYS.Lib, QOpenSys, QOPT, and QFileSvr.400 file systems. It also allows you to transfer folders and documents in the document library services (QDLS) file system. The FTP client can be run interactively in an unattended batch mode where client subcommands are read from a file and the responses to these subcommands are written to a file. It also includes other features for manipulating files on your system.

The client has a user interface from which you can enter client subcommands for making requests to an FTP server. The results of these requests are then displayed.

To transfer files between the client and the server, two connections are established. The control connection is used to request services from the server with FTP server commands. The server sends replies back to

the client to indicate how the request was handled. The second connection, called the data connection, is used for transferring lists of files and the actual file data.

Both the client and the server have a data transfer function that interfaces to the resident file systems. These functions read or write data to the local file systems and to and from the data connection.

## Start and stop a client session

You can start and stop a client session from this topic.

This topic provides details for using the File Transfer Protocol (FTP) client on the iSeries server.

“Start an FTP client session”

“Stop the FTP client session” on page 29

## Start an FTP client session

Before starting the FTP client function, you must have the following information:

- The name or Internet address of the system to which files are sent or obtained.
- A logon ID and password (if required) for the remote system where the file transfers are to occur.
- The name of the file or files with which you want to work (send and receive, for example).

The Start TCP/IP File Transfer Protocol (STRTCPFTP "remotesystem") command starts a client session on the local iSeries server, and then opens a connection to the FTP server on the specified remote system. For example, entering the command FTP myserver.com would start a client session on the iSeries server, and then open a connection to the FTP server on the remote myserver.com system. You can specify additional parameters, or be prompted for them by typing STRTCPFTP without specifying a remote system.

```
Start TCP/IP File Transfer (FTP)

Type choices, press Enter.

Remote system . . . . . > MYSERVER.COM

Coded character set identifier *DFT 1-65533, *DFT
Port . . . . . > *SECURE 1-65535, *DFT, *SECURE
Secure connection . . . . . *DFT *DFT, *NONE, *SSL, *IMPLICIT
Data protection . . . . . *DFT *DFT, *CLEAR, *PRIVATE
```

After you specify a remote system name, you are prompted to specify additional information. The following content summarizes the options available, and additional details that are available in the field help:

### Remote system (RMTSYS)

Specifies the remote system name to which or from which the files are transferred. The following items are possible values:

#### \*INTNETADR

The Internet address (INTNETADR) parameter is prompted. The Internet address is specified in the form, nnn.nnn.nnn.nnn, where nnn is a decimal number ranging from 0 through 255

#### remote-system

Specify the remote system name to which or from which the file transfer takes place.



## Coded character set identifier (CCSID)

Specifies the ASCII coded character set identifier (CCSID) that is used for single-byte character set (SBCS) ASCII file transfers when the FTP TYPE mode is set to ASCII. The possible values are:

**\*DFT** The CCSID value 00819 (ISO 8859-1 8-bit ASCII) is used.

### CCSID-value

The requested CCSID value is used. This value is validated to ensure a valid ASCII SBCS CCSID was requested.

## Port (PORT)

Specifies the port number used for connecting to the FTP server. Normally, the common port value of 21 is used to connect to the FTP server. Under some circumstances, the FTP server can be contacted at a port other than port 21. In those situations, the port parameter can be used to specify the server port to connect to. The possible values are:

**\*DFT** The value 00021 is used.

### \*SECURE

The value 00990 is used. Port 990 is reserved for secure FTP servers which immediately use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) protocols to encrypt data.

### port\_value

The requested port value is used. This value is validated to ensure it is in the proper range.

**Note:** If 990 is specified, the FTP client will perform the same functions as if \*SECURE were specified.

## Secure connection (SECCNN)

Specifies the type of security mechanism to be used for protecting information transferred on the FTP control connection (which includes the password used to authenticate the session with the FTP server). Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are compatible protocols which use encryption to protect data from being viewed during transmission and verify that data loss or corruption does not occur.

**Note:** The FTP client subcommand SECOPEN can be used to open a protected FTP connection during an FTP client session.

The possible values are:

**\*DFT** If the PORT parameter specifies \*SECURE or 990, \*IMPLICIT is used; otherwise, \*NONE is used.

### \*IMPLICIT

The FTP client immediately attempts to use TLS or SSL when connecting to the specified FTP server (without sending an AUTH subcommand to the server). If the server does not use implicit TLS or SSL on the specified port, or the TLS or SSL negotiation fails for any reason, the connection is closed.

**\*SSL** After connecting to the specified FTP server, the FTP client sends an AUTH (authorization) subcommand requesting a TLS or SSL protected session. If the server supports TLS or SSL, a TLS or SSL negotiation performed. If the server does not support TLS or SSL or the TLS or SSL negotiation fails, the connection is closed.

### \*NONE

The FTP client does not use encryption for the control connection to the specified FTP server.

## Data protection (DTAPROT)

Specifies the type of data protection to be used for information transferred on the FTP data connection. This connection is used to transfer file data and directory listings. The FTP protocol does not allow protection of the data connection if the control connection is not protected.

**Note:** The FTP client subcommand SECDATA can be used subsequently to change the data protection level. The FTP client uses the FTP server subcommand PROT to request the specified data protection after a secure control connection has been established.

The possible values are:

**\*DFT** If the SECCNN parameter specifies a protected control connection, \*PRIVATE is used; otherwise, \*CLEAR is used.

**\*PRIVATE**

Information sent on the FTP data connection is encrypted. If the SECCNN parameter specifies that the FTP control connection is not encrypted, \*PRIVATE cannot be specified.

**\*CLEAR**

Information sent on the FTP data connection is not encrypted.

## Outgoing ASCII/EBCDIC table (TBLFTPOUT)

Specifies the table object that is to be used to map all outgoing data in the FTP client. Outgoing data is mapped from EBCDIC to ASCII. If no table object is specified for TBLFTPOUT, the CCSID parameter is used to determine outgoing mapping. The possible values are:

**\*CCSID**

The CCSID parameter is used to determine outgoing mapping.

**\*DFT** The CCSID parameter is used to determine outgoing mapping.

The name of the outgoing mapping table can be qualified by one of the following library values:

**\*LIBL** All libraries in the user and system portions of the job's library list are searched until the first match is found.

**\*CURLIB**

The current library for the job is searched. If no library is specified as the current library for the job, the QGPL library is used.

**library-name**

Specify the name of the library to be searched.

**outgoing-mapping-table**

Specify the table object to be used by the FTP client for mapping outgoing data.

## Incoming ASCII/EBCDIC table (TBLFTPIN)

Specifies the table object that is to be used to map all incoming data in the FTP client. Incoming data is mapped from ASCII to EBCDIC. If no table object is specified for TBLFTPIN, the CCSID parameter is used to determine incoming mapping. The following items are possible values:

**\*CCSID**

The CCSID parameter is used to determine incoming mapping.

**\*DFT** The CCSID parameter is used to determine incoming mapping.

The name of the incoming mapping table can be qualified by one of the following library values:

**\*LIBL** All libraries in the user and system portions of the job's library list are searched until the first match is found.

**\*CURLIB**

The current library for the job is searched. If no library is specified as the current library for the job, the QGPL library is used.

**library-name**

Specify the name of the library to be searched.

**incoming-mapping-table**

Specify the table object to be used by the FTP client for mapping incoming data.

## Stop the FTP client session

Use the QUIT subcommand to stop an FTP session. The QUIT subcommand closes the connection with the remote host and ends the FTP session on the iSeries server. Alternatively, you can press F3 (Exit), and then confirm to end the FTP client session.

**Related concepts**

“Secure the FTP client with Transport Layer Security or Secure Socket Layer” on page 20  
You can use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) connections to encrypt data transferred over File Transfer Protocol (FTP) control and data connections.

**Related tasks**

“Transfer files with File Transfer Protocol”  
You can send and receive files with File Transfer Protocol (FTP).

**Related information**

“Scenario: Transfer a file from a remote host” on page 1  
To transfer a file to a test server, use basic File Transfer Protocol (FTP) to send the file to the remote host.

## Server timeout considerations

You can keep your connection from timing out.

The inactivity time-out value requires some consideration. This is the time in seconds without File Transfer Protocol (FTP) server activity that will cause the server to close the session. Certain remote servers allow the client to change this value. For example, iSeries supports the FTP server TIME subcommand, which can be sent to the server with the FTP client QUOTE subcommand, as described in QUOTE (Send a Subcommand to an FTP Server). UNIX servers often support the SITE IDLE subcommand.

When using local iSeries subcommands with either the SYSCMD subcommand or F21, there is no interaction between the client and the server. Therefore, if the running of these local iSeries commands exceeds the server inactivity time-out period, the server will close the connection. If you lose your connection, you must log on to the server again using the OPEN command (OPEN <remote system name>) and the USER command as described in the note to Logon to the Remote System (Server).

**Related reference**

“QUOTE (Send a Subcommand to an FTP Server)” on page 77

## Transfer files with File Transfer Protocol

You can send and receive files with File Transfer Protocol (FTP).

Follow these steps to transfer files with FTP.

1. Collect the following information:
  - The TCP/IP name or IP address of the remote computer
  - A logon name and password for the remote computer (unless the remote computer supports anonymous FTP)

- The name and location of the file you want to transfer
  - The location of the destination
  - The file transfer type that you use: ASCII, EBCDIC, or BINARY
  - Whether you want to use a connection secured with Transport Layer Security (TLS) or Secure Sockets Layer (SSL).
2. At the command line, type FTP and press Enter.
  3. At the prompt, enter the TCP/IP name or IP address of the remote computer system and press Enter. You can use either the name or the IP address, such as:

```
remote.systemname.com
or
110.25.9.13
```

4. Enter the Coded Character Set Identifier (CCSID). Use the default (\*DFT) value unless you know that you need a specific CCSID.
5. If you want to use a secure connection to protect passwords and data, specify a Port value of \*SECURE.
6. Press Enter to initiate the connection. The FTP client displays messages that indicate a successful connection with the remote system.

**Note:** If you specify a port of \*SECURE and the server does not support implicit TLS or SSL on the specified port, or the TLS or SSL negotiation fails for any reason, the connection is closed.

7. To change the file transfer type, complete the following steps:
  - a. To switch to EBCDIC, enter EBCDIC and press Enter before you transfer the file.
  - b. To switch to BINARY, enter BINARY and press Enter before you transfer the file.
  - c. To switch back to the default type, ASCII, enter ASCII and press Enter before you transfer the file.
8. Now you are ready to transfer files:
  - a. Enter CD and the name of the directory. Press Enter.
  - b. Do one of the following steps:
    - To transfer a file from the server system to the client system, enter GET followed by the name of the file:  
GET myfile.txt
    - To send a file that is on the client system to the server system, enter PUT followed by the name of the file:  
PUT myfile.txt
9. Enter the FTP subcommand QUIT to end the FTP client session and return to the iSeries command line.

#### **Related concepts**

“Data transfer methods” on page 124

Before you begin to transfer files, you must choose the appropriate file transfer type.

#### **Related reference**

“Start and stop a client session” on page 26

You can start and stop a client session from this topic.

#### **Related information**

“Scenario: Transfer a file from a remote host” on page 1

To transfer a file to a test server, use basic File Transfer Protocol (FTP) to send the file to the remote host.

## **Run File Transfer Protocol in unattended mode using a batch job**

This topic provides examples of how to run File Transfer Protocol (FTP) in an unattended mode.

In addition to running the FTP client interactively, you can run the FTP client in an unattended mode. This topic provides two examples of this method: a "Batch FTP: A simple example" and a "Batch FTP: A complex example". You can also refer to Section 6.7, Batch FTP, of V4 TCP/IP for AS/400®: More Cool Things Than Ever (about 10 074 KB) for another example.

**Related concepts**

"Control File Transfer Protocol access" on page 16

If you are using File Transfer Protocol (FTP), you need to control users to protect your data and network. This topic offers tips and security considerations.

**Related information**

"Scenario: Transfer a file from a remote host" on page 1

To transfer a file to a test server, use basic File Transfer Protocol (FTP) to send the file to the remote host.

V4 TCP/IP for AS/400: More Cool Things Than Ever

**Simple example: Batch FTP**

This simple example shows a batch file transfer that involves the successful transfer of one file from a remote system.

The components are as follows:

- A CL program
- An input file of FTP commands
- An output file of FTP messages

**The CL program**

```
*****
ITSOLIB1/QCLSRC BATCHFTP:
-----
      PGM
      OVRDBF  FILE(INPUT) TOFILE(ITSOLIB1/QCLSRC) MBR(FTPCMDS)
      OVRDBF  FILE(OUTPUT) TOFILE(ITSOLIB1/QCLSRC) MBR(OUT)
      FTP     RMTSYS(SYSxxx)
      ENDPGM
*****
```

**Note:** To make this sample work when written with ILECL, you must add OVRSCOPE(\*CALLLVL) to the OVRDBF commands.

The BATCHFTP program overrides the INPUT parameter to the source physical file ITSOLIB1/QCLSRC MBR(FTPCMDS). The output is sent to MBR(OUT).

**The input commands file**

```
*****
ITSOLIB1/QCLSRC FTPCMDS:
-----
      ITSO ITSO
      CD ITSOLIB1
      SYSCMD CHGCURLIB ITSOLIB2
      GET QCLSRC.BATCHFTP QCLSRC.BATCHFTP (REPLACE
      QUIT
*****
```

The FTP subcommands required are shown in the FTPCMDS file.

**The output messages file**

```
*****
      FTP Output Redirected to a File
      FTP Input from Overridden File
*****
```

```

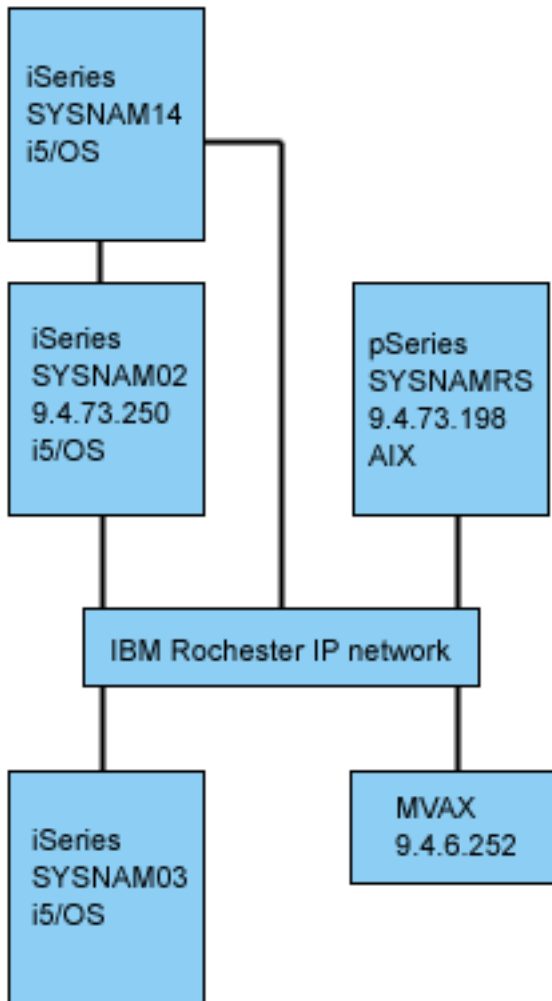
Connecting to host name SYSxxx
at address x.xxx.xx.xxx using port 21.
220-QTCP at SYSxxx.sysnam123.ibm.com.
220 Connection will close if idle more than 5 minutes.
Enter login ID (itso):
> ITSO ITSO
331 Enter password.
230 ITSO logged on.
  i5/OS is the remote operating system. The TCP/IP version is "V3R1M0".
250 Now using naming format "0".
257 "QGPL" is current library.
Enter an FTP subcommand.
> CD ITSOLIB1
Enter an FTP subcommand.
250 Current library changed to ITSOLIB1.
> SYSCMD CHGCURLIB ITSOLIB2
Enter an FTP subcommand.
> GET QCLSRC.BATCHFTP QCLSRC.BATCHFTP (REPLACE
200 PORT subcommand request successful.
150 Retrieving member BATCHFTP in file QCLSRC in library ITSOLIB1.
250 File transfer completed successfully.
147 bytes transferred in 0.487 seconds. Transfer rate 0.302 KB/sec.
Enter an FTP subcommand.
> QUIT
221 QUIT subcommand received.
*****

```

The output file is shown. It is a straightforward matter to write a program to process this file and display an error message on QSYSOPR if there are any error messages. FTP error messages have numbers that start with a 4 or 5.

### **Complex example: Batch FTP**

The following example shows how to retrieve files from several remote hosts to a central iSeries in batch mode.



User GWIL on iSeries SYSNAM03 wants to do the following steps:

1. Retrieve files from hosts SYSNAMRS (RS/6000<sup>®</sup>) and MVAX (VAX).
2. After retrieving the file from SYSNAMRS, the file should be transferred to SYSNAM02 (another iSeries) using FTP.
3. From there the file is to be sent using TCP/IP to iSeries SYSNAM14.

**Example: Create a CL program to start FTP:**

1. As shown in the previous example, FTP uses the display station for command INPUT and message OUTPUT, and this needs to be overridden for use in batch mode. We use the OVRDBF command to overwrite these files with the ones to be used in batch:

```

OVRDBF FILE(INPUT) TOFILE(GERRYLIB/QCLSRC) MBR(FTPCMDS)
OVRDBF FILE(OUTPUT) TOFILE(GERRYLIB/QCLSRC) MBR(FTPLOG)
  
```

2. A host name or an internet address is a required parameter for the STRTCPFTP command that is included in the CL program file. However, if one wants to specify the remote systems in the input commands file instead of the CL program file, then a dummy host name must be specified for the STRTCPFTP command to satisfy the required syntax. This dummy name can be a fictitious host name or a real host name. If it is a real name, then the first entry in the input commands file must be a user ID and a password, and the second entry must be the CLOSE subcommand. If it is not a real host name, then these entries are not required, and the first entry should be an OPEN subcommand to connect to the required server system.

```

FTP RMTSYS(LOOPBACK)
  
```

FTP processes the input file and writes messages to the output file (FTPLOG).

3. After the FTP application ends, delete the overrides:

```
DLTOVR FILE(INPUT OUTPUT)
```

The CL program for batch FTP will look like the following example, in Figure 1, on system SYSNAM01:

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

```
Columns . . . : 1 71          Browse          GERRYLIB/QCLSRC
SEU==>                                     FTPBATCH
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
***** Beginning of data *****
0001.00 PGM
0002.00          OVRDBF  FILE(INPUT) TOFILE(GERRYLIB/QCLSRC) +
0003.00          MBR(FTPCMD5)
0004.00          OVRDBF  FILE(OUTPUT) TOFILE(GERRYLIB/QCLSRC) +
0005.00          MBR(FTPLOG)
0006.00          FTP     RMTSYS(LOOPBACK) /* (FTP CL Program) */
0007.00          DLTOVR  FILE(INPUT OUTPUT)
0008.00 ENDPGM
***** End of data *****

F3=Exit  F5=Refresh  F9=Retrieve  F10=Cursor  F12=Cance
F16=Repeat find      F24=More keys

(C) COPYRIGHT IBM CORP. 1981, 1994.
```

Figure 1. CL program FTPBATCH for batch FTP

**Example: Create the FTP input file (FTCPDMS):**

The FTP input file must contain all the FTP client subcommands necessary to connect and log on to the server, set up and complete the file transfers, close the server connection, and end the client session. The following example shows the subcommands used for transferring files to two different remote systems.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.



```

Columns . . . :   1 71          Browse          GERRYLIB/QCLSRC
SEU==>          FTPCMDS
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5...+... 6 ...+... 7
          ***** Beginning of data*****
0001.00 gwil ****
0002.00 close
0003.00 open sysnamrs
0004.00 user root root
0005.00 ascii
0006.00 syscmd dlrf file(gerrylib/rs6)
0007.00 get /Itsotest gerrylib/rs6.rs6
0008.00 close
0009.00 open mvax
0010.00 user tester tester
0011.00 get screen1.file gerrylib/vax.vax (replace
0012.00 close
0013.00 open sysnam02
0014.00 user gwil ****
0015.00 ebcidc
0016.00 put gerrylib/rs6.rs6 gerrylib/rs6.rs6
0017.00 quote rcmd sndnetf file(gerrylib/rs6) tousrid((gwilsysnam14))
0018.00 close
0019.00 quit
          ***** End of data*****
F3=Exit  F5=Refresh  F9=Retrieve  F10=Cursor  F12=Cancel
F16=Repeat find      F24=More keys

```

Figure 2. Transferring files to two remote systems

The following explanation might help you to understand the FTP client subcommands shown in Figure 2. The line numbers on the display correspond to the numbers that follow.

- 0001 User ID and password for dummy connection within client iSeries SYSNAM03.
- 0002 Close dummy connection in iSeries SYSNAM03.
- 0003 Open control connection to RISC System/6000® SYSNAMRS.
- 0004 USER subcommand with user ID and password for SYSNAMRS.

**Note:** When running FTP in batch mode, the USER subcommand must follow an OPEN subcommand. Both the logon user ID and password parameters for the USER subcommand should be provided. This is different when operating FTP interactively online. When FTP is run interactively online, and then the client will automatically initiate a USER subcommand and prompt you for a logon ID. There is no automatic USER subcommand when running FTP in batch mode.

- 0005 Transfer ASCII data (will be converted on iSeries to/from EBCDIC).
- 0006 CL command to be run on client iSeries: delete file. Instead parameter (REPLACE can be used with the next statement.)
- 0007 Retrieve file from RISC System/6000 system.
- 0008 Close control connection to RISC System/6000 SYSNAMRS.
- 0009 Open connection to VAX MVAX.
- 0010 USER subcommand with user ID and password for MVAX.
- 0011 Retrieve file from VAX replacing existing iSeries file.
- 0012 Close control connection to VAX MVAX.
- 0013 Open control connection to remote iSeries SYSNAM02.

- 0014 USER subcommand with user ID and password for SYSNAM02.
- 0015 Transfer EBCDIC data (as it is from iSeries to iSeries).
- 0016 Send iSeries file to iSeries SYSNAM02 with TCP/IP.
- 0017 Send this file from server iSeries SYSNAM03 to remote iSeries SYSNAM14 through TCP/IP network.
- 0018 Close control connection to iSeries SYSNAM02.
- 0019 End FTP application.

**Example: CL program for submitting the FTPBATCH job:**

To schedule the file transfers and run them in unattended mode, create a CL program that submits the FTPBATCH job. In the following example, the file transfers are to run the next Friday, 17:00 hour, in unattended mode.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

```

Columns . . . : 1 71          Browse          GERRYLIB/QCLSRC
SEU==>          FTPSUBMIT
FMT ** ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5...+... 6 ...+... 7
***** Beginning of data*****
0001.00 PGM
0002.00          SBMJOB      CMD(CALLPGM(GERRYLIB/FTPBatch)) +
0003.00          JOB(FTPFRIDAY)OUTQ(QUSRSYS/GERRYQ)      +
0004.00          SCDDATE(*FRI)SCDTIME(170000) /* FTP for +
0005.00          Friday, 5:00 in theafternoon */
0006.00 ENDPGM
***** End of data*****

F3=Exit  F5=Refresh  F9=Retrieve  F10=Cursor  F12=Cancel
F16=Repeat find      F24=More keys

(C) COPYRIGHT IBM CORP. 1981, 1994.

```

Figure 3. CL program for submitting batch FTP job

**Example: Check the FTP output file for errors:**

While running at the scheduled time, FTP creates the data in file member FTPLOG shown in figure 3. The data in file member FTPLOG corresponds to original statements found in both examples.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

```

    Connecting to host name LOOPBACK at address 127.0.0.1 using port 21.
220-QTCP at localhost.
220 Connection will close if idle more than 5 minutes.
Enter login ID (gwil):

>>>GWIL ****
331 Enter password.
230 GWIL logged on.
    i5/OS is the remote operating system. The TCP/IP version is "V4R2M0".
250 Now using naming format "0".
257 "QGPL" is current library.
Enter an FTP subcommand.

> CLOSE
221 QUIT subcommand received.
Enter an FTP subcommand.

> OPEN SYSNAMRS
Connecting to host name SYSNAMRS at address 9.4.73.198 using port 21.
220 sysnamrs.sysnam123.ibm.com FTP server (Version 4.9 ThuSep 2 20:35:07 CDT
    1993) ready.
Enter an FTP subcommand.

```

Figure 4. FTP output (FTPLOG) after running FTPBATCH program (Part 1 of 5)

```

> USER root ****
331 Password required for root.
230 User root logged in.
UNIX Type: L8 Version: BSD-44
Enter an FTP subcommand.

> ASCII
200 Type set to A; form set to N.
Enter an FTP subcommand.

> SYSCMD DLT FILE(GERRYLIB/RS6)
Enter an FTP subcommand.

> GET /Itsotest GERRYLIB/RS6/RS7
200 PORT command successful.
150 Opening data connection for /Itsotest (467 bytes).
226 Transfer complete.
467 bytes transferred in 2.845 seconds. Transfer rate 0.167KB/sec.
Enter an FTP subcommand.

```

Figure 5. FTP output (FTPLOG) after running FTPBATCH program (Part 2 of 5)

```

> CLOSE
221 Goodbye.
Enter an FTP subcommand.

> OPEN MVAX
Connecting to host system mvax at address 9.4.6.252 usingport 21.
220 FTP Service Ready
Enter an FTP subcommand.

> USER TESTER *****
331 User name TESTER received, please send password
230 TESTER logged in, directory $DISK1:[TESTER]
Enter an FTP subcommand.

GET SCREEN1.FILE GERRYLIB/VAX.VAX (REPLACE
200 PORT Command OK.
125 ASCII transfer started for $DISK1:[TESTERSCREEN1.FILE;1(266586 bytes)
226 File transfer completed ok.
265037 bytes transferred in 8.635 seconds. Transfer rate30.694 KB/sec.
Enter an FTP subcommand.

> CLOSE
221 Goodbye.
Enter an FTP subcommand.

OPEN SYSNAM02
Connecting to host system SYSNAM02 at address 9.4.73.250using port 21.
220-QTCP at SYSNAM02.sysnam123.ibm.com.
220 Connection will close if idle more than 5 minutes.
Enter an FTP subcommand.

```

Figure 6. FTP output (FTPLOG) after running FTPBATCH program (Part 3 of 5)

```

> USER GWIL ****
331 Enter password.
230 GWIL logged on.
i5/OS is the remote operating system. The TCP/IP version is "V4R2M0".
250 Now using naming format "0".
257 "QGPL" is current library.
Enter an FTP subcommand.

> EBCDIC
200 Representation type is EBCDIC nonprint.
Enter an FTP subcommand.

> PUT GERRYLIB/RS6.RS6 GERRYLIB/RS6.RS6
200 PORT subcommand request successful.
150 Sending file to member RS6 in file RS6 in library GERRYLIB.
250 File transfer completed successfully.
467 bytes transferred in 0.148 seconds. Transfer rate 3.146 KB/sec.
Enter an FTP subcommand.

> RCMD SNDNETF FILE(GERRYLIB/RS6) TOUSRID((GERRYLIB SYSNAM14))
250 Command SNDNETF FILE(GERRYLIB/RS6) TOUSRID((GWIL SYSNAM14))
successful.
Enter an FTP subcommand.

```

Figure 7. FTP output (FTPLOG) after running FTPBATCH program (Part 4 of 5)

```

> CLOSE
221 QUIT subcommand received.
Enter an FTP subcommand.
> QUIT
(This ends the FTP application)

```

Figure 8. FTP output (FTPLOG) after running FTPBATCH program (Part 5 of 5)

Check the output for errors that might have occurred during FTP processing. You can either check visually or run a program that tests for error reply codes. Three-digit FTP error reply codes start with 4 or 5. Be careful to avoid messages, such as '467 bytes transferred...'.

*Sample procedure:* A sample REXX procedure and a sample physical file member are shipped as part of the TCP/IP product. File QATMPINC in library QTCP includes the following two members:

- BATCHFTP that contains REXX source code to specify the input and output batch files, and start FTP.
- BFTPFILE that contains the subcommands and data required for logon and running FTP.

## File Transfer Protocol reference information

You can find information about server and client subcommands, File Transfer Protocol (FTP) exit programs, data transfer methods, and more.

### File Transfer Protocol server subcommands

These subcommands represent communication between the client and server. This topic includes descriptions for iSeries CL-equivalent subcommands that are unique to iSeries FTP server.

This topic describes FTP server subcommands. The FTP client communicates with the server using server subcommands. This topic provides the server subcommands, descriptions of what they do, their syntax conventions, and FTP reply status messages.

iSeries FTP server uses the subcommands listed in the following table.

Subcommand	What it does
ABOR	Cancels the previous subcommand
ADDM	Adds a member to a physical file
ADDV	Adds a member to a variable-length member to a physical file
APPE	Appends data to a specified file
AUTH	Defines the authentication mechanism used for the current FTP session
CDUP	Changes directory to the parent directory
CRTL	Creates a library
CRTP	Creates a physical file
CRTS	Creates a source physical file
CWD	Changes the working directory or library
DEBUG	Starts or ends a server trace
DELE	Deletes a file, a member, or a document
DLTF	Deletes a file

Subcommand	What it does
DTL	Deletes a library
HELP	Gets information about FTP server subcommands
LIST	Lists files or directory entries
MKD	Makes a directory
MODE	Specifies a format for data transmission
NLST	Lists the names of files or directories
NOOP	Checks if server is responding
PASS	Sends a password to the server
PASV	Tells the server to passively open the next data connection
PBSZ	Defines the largest buffer protection buffer size to be used for application-level encoded data sent or received on the data connection
PORT	Identifies the data port on which the client will listen for a connection
PROT	Defines the protection used for FTP data connections
PWD	Displays the current working directory
QUIT	Logs off the user; closes the connection
RCMD	Sends a CL command to an FTP server
REIN	Re-starts a session on a server
RETR	Retrieves data from a server
RMD	Removes a directory
RNFR	Specifies a file to be renamed
RNTO	Specifies a new file name
SITE	Sends information for a server to use
STAT	Gets status information from a server
STOR	Saves data on a server and replaces an existing file
STOU	Saves data on a server but does not replace an existing file
STRU	Specifies the structure of a file
SYST	Prints the name of the OS on the server
TIME	Sets the time-Out value for the FTP server
TYPE	Specifies the file transfer type
USER	Sends a user logon ID to the server
XCUP	Changes to the parent directory
XCWD	Changes to the working directory
XMKD	Creates a directory
XPWD	Displays the current directory or library
XRMD	Removes a directory

## Subcommands unique to iSeries FTP server

iSeries FTP server subcommands include a special set of commands that are abbreviated names of equivalent, but longer, iSeries CL commands.

The names of these special server subcommands must be four characters in length to comply with the FTP limits. When the iSeries server receives these subcommands, the name are interpreted in this way:

- ADDM = ADDPFM (Add Physical File Member)
- ADDV = ADDPVLM (Add Physical File Variable Length Member)
- CRTL = CRTLIB (Create Library)
- CRTP = CRTPF (Create Physical File)
- CRTS = CRTSRCPF (Create Source Physical File)
- DLTF = DLTF (Delete File)
- DLTL = DLTLIB (Delete Library)

In addition to these specific subcommands, you can use the FTP server subcommand RCMD to send any CL command to the server.

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

“File Transfer Protocol server reply status messages” on page 134

This topic contains access common reply codes and what they indicate.

“File Transfer Protocol client subcommands” on page 57

Use File Transfer Protocol (FTP) client subcommands to establish a connection with a remote FTP server, navigate libraries and directories, create and delete files, and transfer files.

## ADDM (Add Physical File Member) FTP server subcommand

ADDM parameters
-----------------

### parameters

The parameters for this subcommand are the same as for the ADDPFM CL command.

For example, to add member BANANA to physical file GEORGE in library RLKAYS on an iSeries, enter this:

```
ADDM FILE(RLKAYS/GEORGE) MBR(BANANA)
```

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## ADDV (Add Physical File Variable Length Member) FTP server subcommand

ADDV parameters
-----------------

### parameters

The parameters for this subcommand are the same as for the ADDPVLM CL command.

For example, to add member POLEBEAN to physical file GEORGE in library RLKAYS on an iSeries, enter this:

```
ADDV FILE(RLKAYS/GEORGE) MBR(POLEBEAN)
```

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **ADDV (Add Physical File Variable Length Member) FTP server subcommand**

ADDV parameters
-----------------

**parameters**

The parameters for this subcommand are the same as for the ADDPVLM CL command.

For example, to add member POLEBEAN to physical file GEORGE in library RLKAYS on an iSeries, enter this:

```
ADDV FILE(RLKAYS/GEORGE) MBR(POLEBEAN)
```

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **APPE (Append to Existing File) FTP server subcommand**

The APPE FTP server subcommand accepts the transferred data and stores it in a file on the server system. If the file specified exists, it appends the data to that file; otherwise, it creates the specified file.

APPE <i>filename</i>
----------------------

**filename**

The file that will receive your on the server system.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **AUTH (Authorization) FTP server subcommand**

The AUTH FTP server subcommand defines the authentication/security mechanism that is used for the current FTP session. The syntax of this subcommand is:

```
AUTH [ TLS-C | TLS-P | TLS | SSL ]
```



Parameter values:

TLS-C	Utilize the Transport Layer Security (TLS) protocol as the security mechanism. The security settings for the data connection use the RFC2228 defaults; that is, there is no implicit protection of the data connection.
TLS-P	Utilize the TLS protocol as the security mechanism. Also, implicitly protect the data connection (which is equivalent to the command sequence AUTH TLC-C, PBSZ 0, PROT P)
TLS	Synonym for TLS-C.
SSL	Synonym for TLS-P.

**Note:** The TLS protocol is compatible with the Secure Sockets Layer (SSL) protocol.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **CRTL (Create Library) FTP server subcommand**

CRTL parameters

**parameters**

The parameters for this subcommand are the same as for the CRTLIB CL command.

For example, to create a library that is called TESTTCP on an iSeries server, enter this:

```
CRTL TESTTCP
```

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **CRTP (Create Physical File) FTP server subcommand**

CRTP parameters

**parameters**

The parameters for this subcommand are the same as for the CRTPF CL command.

For example, to create a physical file that is called MYFILE with a record length of 80 and no restrictions on the number of members, enter this:

```
CRTP FILE(RLKAYS/MYFILE) RCDLEN(80) MAXMBRS(*NOMAX)
```

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **CRTS (Create Source Physical File) FTP server subcommand**

CRTS parameters

### **parameters**

The parameters for this subcommand are the same as for the CRTSRCPF CL command.

For example, to create a source physical file that is called GEORGE in library RLKAYS, enter this:

```
CRTS FILE(RLKAYS/GEORGE)
```

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **CWD (Change Working Directory or Library) FTP server subcommand**

To change the working directory, library, or file group, use the CWD FTP server subcommand.

CWD *directory*

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **DEBUG (Turn on the FTP Server Trace) FTP server subcommand**

**Note:** Use the FTP server trace only for reporting software problems to IBM. You might affect system performance by this function.

DEBUG

If the FTP server trace is not active, the server starts a trace. The server continues to run a trace until it receives another DEBUG subcommand or a QUIT subcommand. When it ends the trace, there might be a significant delay while it formats the trace data.

### **Related concepts**

“Trace the FTP server” on page 146

You can use the DEBUG subcommand to track down problems on the FTP server.

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **DELE (Delete File or Document) FTP server subcommand**

To delete a file, a member, or a document, use the CWD FTP server subcommand.

```
DELE remotefile
```

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **DLTF (Delete File) FTP server subcommand**

```
DLTF parameters
```

### **parameters**

The parameters for this subcommand are the same as for the DLTF CL command.

For example, to delete file MYFILE in library RLKAYS, enter this:

```
DLTF FILE(RLKAYS/MYFILE)
```

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **DLTL (Delete Library) FTP server subcommand**

```
DLTL parameters
```

### **parameters**

The parameters for this subcommand are the same as for the DLTLIB CL command.

For example, to delete a library, enter this:

```
DLTL libname
```

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **HELP (Getting Help from an iSeries Remote Server) FTP server subcommand**

To get information about the FTP server subcommands, use the HELP subcommand in this format:

```
HELP [subcommand]
```

### subcommand

The name of the server subcommand you want information about. For example, HELP ADDM will provide help information about how to add a member to a physical file on an iSeries.

To determine the syntax of the ADDV subcommand that is used by the iSeries server, use the server subcommand:

```
HELP ADDV
```

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## LIST (File List) FTP server subcommand

To get a list of directory entries, library contents, or files in a file group, use the LIST FTP server subcommand:

```
LIST [directory | name]
```

It lists only those files that FTP can transfer

### Related tasks

“Configure FTP servers for graphical FTP clients and Web tools” on page 8

You can configure an File Transfer Protocol (FTP) server on the iSeries to support graphical FTP clients, Web browsers, and other Web tools.

### Related reference

“Use the LIST subcommand information in iSeries format” on page 8

“LIST information in UNIX-style format” on page 9

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

“SITE (Send Information Used by a Server System)” on page 52

## MKD (Make Directory) FTP server subcommand

To create or make a directory, use the MKD FTP server subcommand.

```
MKD directoryname
```

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## MODE (Set Transfer Mode) FTP server subcommand

To specify how to you want bits of data transmitted, specify the mode, or data format, by using the MODE FTP server subcommand:

```
MODE [B | S]
```

- B** Specifies block mode. In this mode, data is a series of data blocks, preceded by one or more header bytes.
- S** Specifies stream mode. In this mode, data a stream of bytes. You can use any representation type with stream mode. This transfer mode is more efficient because the server does not transfer any data block information.

**Notes:**

1. Stream mode is the default transfer mode the iSeries server uses and is the preferred mode.
2. If there is no parameter, the server returns a reply that indicates the present setting for MODE.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **NLST (Name List) FTP server subcommand**

To get a list of only the names of multiple files, a file group, a directory, or a library, use the NLST FTP server subcommand:

```
NLST [directory | name]
```

It lists only those files that FTP can transfer.

**Related tasks**

“Configure FTP servers for graphical FTP clients and Web tools” on page 8

You can configure an File Transfer Protocol (FTP) server on the iSeries to support graphical FTP clients, Web browsers, and other Web tools.

**Related reference**

“Use the LIST subcommand information in iSeries format” on page 8

“LIST information in UNIX-style format” on page 9

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **NOOP (Obtain Server Response) FTP server subcommand**

The NOOP FTP server subcommand sends an “OK” reply to the client. It does not affect server processing in any other way. The client uses this command to determine if the server is connected and responding. Use the NOOP subcommand:

```
NOOP
```

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **PASS (Password) FTP server subcommand**

```
PASS password
```

### **password**

A string that specifies your password for the server system.

**Note:** The USER server subcommand must immediately precede the server subcommand PASS immediately.

#### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **PASV (Use Passive Data Connection) FTP server subcommand**

To instruct this server to passively open the next data connection, use the PASV FTP server subcommand in this format:

```
PASV
```

#### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **PBSZ (Protection Buffer Size) FTP server subcommand**

The PBSZ subcommand defines largest buffer size to be used for application-level encoded data sent or received on the data connection. The syntax of this subcommand is:

```
PBSZ value
```

where *value* is an ASCII character string representing a decimal integer.

**Note:** You must specify a value of '0' for this parameter.

#### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **PORT (Data Port) FTP server subcommand**

To identify the port on which the client will listen for a data connection, use the PORT FTP server subcommand in this format:

```
PORT h1,h2,h3,h4,p1,p2
```

**h n** Represents the system IP address and is a character string that is a decimal value between 0 and 255.

**p n** Represents the TCP port number and is a character string that is a decimal value between 0 and 255.

To convert the p1 and p2 values to a TCP port number, use this formula:

$$\text{port} = ( p1 * 256 ) + p2$$

For example, in this PORT subcommand:

```
PORT 9,180,128,180,4,8
```

the port number is 1032 and the IP address is 9.180.128.180.

**Note:** After the server closes the connection, it cannot connect to the same client IP address and port number until a two-minute time delay has occurred as specified in TCP/IP RFC 1122. The server can make a connection to the same client IP address on a different port number without this restriction.

#### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## PROT (Data Channel Protection Level) FTP server subcommand

The PROT subcommand defines the protection used for FTP data connections (which are used to transmit directory listings and file data). The syntax of this subcommand is:

```
PROT [ C | P ]
```

Parameter values:

C	Clear. The data connection carries “raw data” of the file transfer with no security applied.
P	Private. The data connection will use Transport Layer Security (TLS) or Secure Socket Layer (SSL), which provides Integrity and Confidentiality protection.

#### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## PWD (Display Working Directory or Library) FTP server subcommand

The server returns a reply to the client with the name of the current directory or library when the PWD FTP server subcommand:

PWD

#### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

### QUIT (End an FTP Server Session) FTP server subcommand

The FTP server subcommand QUIT logs off the client user and closes the control connection. If a file transfer is in progress, the connection remains open until the file transfer is complete, and then the server closes it.

QUIT

#### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

### RCMD (Send a CL Command to an FTP Server System) FTP server subcommand

Use the server subcommand RCMD to run iSeries control language (CL) commands on the FTP server system. The length of the RCMD subcommand string is up to 1000 characters. Because no prompting is available for the RCMD subcommand, the RCMD subcommand string must include all necessary parameters to run the CL command.

If the CL command called through the RCMD subcommand runs successfully, a message stating that the subcommand is successful is displayed. If an error occurs, it displays a message that states there was an error. The message does not include what the error was unless the error occurred because a library, file, or member name was not valid.

This is an example of using RCMD to run a Delete File (DLTF) command:

```
QUOTE RCMD DLTF FILE(mylib/myfile)
```

`mylib` is the name of the library from which the file is to be deleted. `myfile` is the name of the file to be deleted.

You might also be interested in reading about the REXEC server, which provides an alternative method for executing CL commands on a remote system.

#### Related concepts

REXEC server

#### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

“QUOTE (Send a Subcommand to an FTP Server)” on page 77



## **REIN (Reinitialize Session between Systems) FTP server subcommand**

```
REIN
```

The REINITIALIZE subcommand:

1. Allows the completion of any transfer in progress
2. Ends the USER session and removes all input/output and account information
3. Resets all server parameters to the default settings
4. Leaves the control connection open

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **RETR (Retrieve File) FTP server subcommand**

To retrieve data from the server system, use the RETR FTP server subcommand.

```
RETR remotefile
```

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **RMD (Remove Directory) FTP server subcommand**

To remove a directory, use the RMD FTP server subcommand.

```
RMD directoryname
```

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **RNFR (Rename From) FTP server subcommand**

The RNFR FTP server subcommand renames files. It must be immediately followed by a RNTO (Rename To) server subcommand.

```
RNFR filename
```

### **filename**

The name of the file you want renamed.

**Note:** The iSeries server cannot rename a file to a different file system.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **RNTO (Rename To) FTP server subcommand**

The RNTO FTP server subcommand specifies the new file name when renaming files on the server system. It must immediately follow an RNFR subcommand, which specified the file name.

RNTO filename
---------------

**filename**

The name to which the file you want renamed.

**Note:** The iSeries server, cannot rename a file to a different file system.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **SITE (Send Information Used by a Server System) FTP server subcommand**

To send information that is used by the server system or to provide services specific to the server system, use the SITE FTP server subcommand in this format:

SITE [ <i>parameters</i> ]
----------------------------

iSeries FTP server supports these parameters for the SITE subcommand:

**LISTFMT 0**

The server returns information for the LIST subcommand in LIST Information in iSeries format, which was introduced in V3R1M0. The iSeries client supports both the iSeries format and the UNIX-style format.

**LISTFMT 1**

The server returns information for the LIST subcommand in LIST information in UNIX-style format. The name of the file is the last item of each line returned. The iSeries client supports both the iSeries format and the UNIX-style format.

**LISTFMT**

Return a message that indicates the current FTP server LISTFMT setting.

**Notes:**

If you want to change the LISTFMT default on the server, then use the LISTFMT option of the CHGFTP command. You can also use iSeries Navigator to set this FTP server property:

1. In iSeries Navigator, expand **your iSeries server** → **Network** → **Servers** → **TCP/IP**.
2. In the right pane, right-click **FTP** and select **Properties**.

3. Click the **Initial Formats** tab.
4. Under the **File List** heading, enable iSeries or UNIX as the LISTFMT default on the server.
5. Click **OK** to accept the changes.

#### **NAMEFMT 0**

Use the LIBRARY/FILE.MEMBER name format. This name format is only for library file system database files.

#### **NAMEFMT 1**

Use the path name format. This name format is for all file systems that are supported by FTP that includes the library file system. Name format 1 must be used to work with all iSeries file systems other than the library file system.

#### **NAMEFMT**

Return a message that contains the current server file name format.

**Note:** You can configure the iSeries FTP server the default NAMEFMT setting with the NAMEFMT option of the CHGFTP command.

#### **CRTCCSID \*CALC**

New database files created during ASCII file transfers use the related default EBCDIC CCSID of the ASCII file transfer CCSID.

#### **CRTCCSID \*USER**

New database files created during ASCII file transfers use the current job CCSID. If this CCSID is 65535, the default CCSID is determined by the language id that specifies the current job.

#### **CRTCCSID \*SYSVAL**

New database files created during ASCII file transfers use the CCSID that was specified by the QCCSID system value.

#### **CRTCCSID [CCSID-number]**

Specify the CCSID when creating database files on the client during ASCII file transfers. The server validates this value.

#### **CRTCCSID**

Display a message that contains the current FTP client CRTCCSID setting.

#### **NULLFLDS 0**

The server does not allow transfer of database files that contain NULL fields. This is the default.

#### **NULLFLDS 1**

The server allows transfer of database files that contain NULL fields.

**Note:** Transfer of files that contain NULL fields requires both the client and server to have this setting enabled. If the server transfers a file that contains NULLfields to a non-iSeries server, or if the transfer type results in codepage conversion of the data, then results are unpredictable.

#### **NULLFLDS**

Return a message that indicates the current FTP server NULLFLDS setting.

#### **TRIM 0**

Set Trim option to OFF. The server sends trailing blanks of database records.

#### **TRIM 1**

Set Trim option to ON. The server does not send trailing blanks of database records when transferring database files that use file structure and stream mode. This is the default.

## TRIM 2

The server does not send trailing blanks of database records for all transfers, including record structure and block mode.

**TRIM** Returns a message that indicates the current setting of the FTP server Trim option.

### Notes:

1. Prior to the availability of this subcommand, trailing blanks of QSYS.LIB file system records were always removed before transferring the file to the server system.
2. TRIM settings do not apply to TYPE I (binary) file transfers. Blanks are never trimmed for TYPE I file transfers, regardless of the TRIM setting.

### Related tasks

“Configure FTP servers for graphical FTP clients and Web tools” on page 8

You can configure an File Transfer Protocol (FTP) server on the iSeries to support graphical FTP clients, Web browsers, and other Web tools.

### Related reference

“Use the LIST subcommand information in iSeries format” on page 8

“LIST information in UNIX-style format” on page 9

“LIST (File List)” on page 46

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## STOR (Store File) FTP server subcommand

To save data on the server system and overlay an existing file, use the STOR FTP server subcommand in this format:

```
STOR remotefile
```

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## STOU (Store Unique) FTP server subcommand

To save data on the server system and not overlay an existing file, use the STOU FTP server subcommand:

```
STOU remotefile
```

The server generates a unique file name. The name assigned to the file will appear in the reply that is sent back to the client.

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **STRU (Specify File Structure) FTP server subcommand**

To specify the structure of a file as a continuous sequence of data bytes, use the STRU FTP server subcommand in this format:

```
STRU [F | R]
```

- F** A file structure. The file structure is a continuous sequence of data bytes.
- R** A record structure. The file is a sequence of sequential records.

### **Notes:**

1. The file structure affects the transfer mode and the interpretation and storage of a file.
2. If there is no parameter, the server returns a reply that indicates the present specification for file structure.

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **SYST (Identify the Name of the Operating System) FTP server subcommand**

To obtain the name of the operating system on the server system, use the SYST FTP server subcommand:

```
SYST
```

The returned information is system dependent.

iSeries server includes the TCP/IP version.

Here is an example server reply:

i5/OS is the remote operating system. The TCP/IP version is "V4R4M0".

### **Related reference**

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## **TIME (Set Time-Out Values for FTP Server) FTP server subcommand**

After the FTP control connection is established between the FTP client and the FTP server, the FTP server controls the time-out for this connection. This is the inactivity time-out value.

There is also a time-out value for the data connection, known as the transfer time-out.

The format of the TIME FTP server subcommand is:

```
TIME inactivity [transfer]
```

## inactivity

The number of seconds the server waits before ending the connection with the client. Inactivity time-out values can range from 1-9,999,999 seconds. The default inactivity time-out value is 300 seconds.

## transfer

The file transfer time-out in seconds. This parameter is optional. If you do not specify this parameter, then the server does not change the current value. Transfer time-out values can range from 1-9,999,999 seconds. The default transfer time-out value is 420 seconds.

For example, to set the inactivity time-out value of the FTP server to 1000 seconds, and keep the current value of the transfer time-out, enter this:

```
QUOTE TIME 1000
```

The TIME subcommand is not a standard FTP subcommand. It is iSeries FTP server specific.

### Related reference

“File Transfer Protocol server syntax conventions” on page 136

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

## TYPE (Specify Representation Type) FTP server subcommand

To specify the file-transfer type or the representation in which the transfer is to take place, use the TYPE FTP server subcommand in this format:

```
TYPE [ A  
      B [ 1 | 2 | 3 [A|R] | 4 [A|R ] | 5 | 6 | 7 ]  
      C ccsid#  
      E  
      F [ 1 ]  
      I ]
```

**A** Specifies the transfer type as the default (ASCII) transfer type. The server does not associate any vertical format control with the file. The server only supports the default format NON PRINT for ASCII. The ASCII transfer type is for the transfer of text files, except when both systems use the EBCDIC type.

**Note:** The CCSID for TYPE A is the CCSID value of the FTP server configuration attributes. You can change these attributes with the CHGFTP command.

**B** Shift JIS Kanji (CCSID 932)

**B 1** Shift JIS Kanji (CCSID 932)

**B 2** Extended UNIX Code Kanji (CCSID 5050)

**B 3** JIS 1983 using ASCII shift-in escape sequence (CCSID 5054)

**B 3 A** JIS 1983 using ASCII shift-in escape sequence (CCSID 5054)

**B 3 R** JIS 1983 using JISROMAN shift-in escape sequence (CCSID 5052)

**B 4** JIS 1978 using ASCII shift-in escape sequence (CCSID 5055)

**B 4 A** JIS 1978 using ASCII shift-in escape sequence (CCSID 5055)

**B 4 R** JIS 1978 using JISROMAN shift-in escape sequence (CCSID 5053)

**B 5** Hangeul (CCSID 934)

- B 6** Korean Standard Code KSC-5601, 1989 version (CCSID 949)
- B 7** Traditional Chinese (5550) (CCSID 938)
- C** Specifies the transfer type to any CCSID (coded character set identifier) that is installed on the system. The CCSID number must follow C.
- E** Specifies the transfer type as EBCDIC. The server does not associate any vertical format control with the file. The server supports only the default format NON PRINT for EBCDIC. The EBCDIC transfer type is for efficient transfer between systems that use EBCDIC for their internal character representation.
- F** IBM EBCDIC Kanji (CCSID 5035)
- F 1** IBM EBCDIC Kanji (CCSID 5035)
- I** Specifies the transfer type as image. With the image transfer type, data is a string of bits, packed into 8-bit bytes. The image transfer type efficiently stores and retrieves files and transfers binary data such as object code.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136  
 The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

**USER (Send a User Logon ID to the Server)  
 FTP server subcommand**

USER username

**username**

The user profile on an iSeries server.

**Note:** If the USER subcommand is successful and the iSeries server is configured for password security, then the server sends a reply to the client requesting a password. The client sends the password to the server with the server subcommand PASS. This is no password prompt when the server is running at security level 10.

**Related reference**

“File Transfer Protocol server syntax conventions” on page 136  
 The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

**File Transfer Protocol client subcommands**

Use File Transfer Protocol (FTP) client subcommands to establish a connection with a remote FTP server, navigate libraries and directories, create and delete files, and transfer files.

File Transfer Protocol (FTP) subcommands instruct the FTP client to transfer files from one computer to another. With FTP client subcommands, you can establish a connection with a remote FTP server, navigate libraries and directories, create and delete files, and transfer files.

You can access descriptions about client subcommands and their syntax from the following topics.

The iSeries FTP client uses these subcommands. The following table identifies client subcommands, the abbreviations, and a description of each subcommand.

Subcommand	Meaning
?	Describes how to use FTP

<b>Subcommand</b>	<b>Meaning</b>
ACCT	Sends a user's account information to a remote system
APPEND	Adds a local file member to file on a remote system
ASCII	Sets the file transfer type to ASCII format
BINARY	Sets the file transfer type to BINARY format
CD	Changes the working directory on remote system
CDUP	Changes to the parent directory on remote system
CLOSE	Ends a session with the remote system
DEBUG	Turns debugging on or off
DEBUG	Changes client time-out values
DELETE	Deletes a file on the remote system
DIR	Displays directories and files on the remote system
EBCDIC	Sets the file transfer type to EBCDIC format
GET	Copies a file from the remote to a local system
HELP	Gets information about FTP client subcommands
LCD	Changes the working directory on local system
LOCSITE	Specifies local site information
LOCSTAT	Displays local status information
LPWD	Displays the working directory on local system
LS	Lists the names of files in a file set on remote system
LTYPE	Specifies the file transfer type on the local system
MDELETE	Deletes multiple files on the server system
MGET	Copies file or files from the remote system
MKDIR	Creates a directory or subdirectory
MODE	Specifies a data format for file transfer
MPUT	Sends local file or files to the remote system
NAMEFMT	Specifies a file naming format to use
NOOP	Checks for a response
NULLFLDS	Allows for NULL fields
OPEN	Connects to an FTP server
PASS	Sends a user's password
PUT	Copies a local file member to remote system
PWD	Displays the current directory of remote system
QUIT	Ends an FTP session
QUOTE	Sends a subcommand to an FTP server
REINITIALIZE	Re-starts a session on a remote system
RENAME	Renames a file on a remote system
RESET	Clears the server reply queue
RMDIR	Removes a directory on the remote system
SECDATA	Specifies the protection level used for the data connection when there is a secure connection established with an FTP server



Subcommand	Meaning
SECOPEN	Opens a secure control connection to an FTP server using the specified security protocol
SENDPASV	Specifies whether a PASV subcommand is sent
SENDPORT	Specifies whether a PORT subcommand is sent
SENDSITE	Specifies whether a SITE subcommand is sent
SITE	Sends information for use by a remote system
STATUS	Gets status information from a remote system
STRUCT	Specifies the file structure of data being sent
SUNIQUE	Controls file replacement
SYSCMD	Runs a CL command on a local system without quitting FTP
SYSTEM	Displays the OS on the remote system
TYPE	Specifies the file transfer type
USER	Sends a user ID to a remote system
VERBOSE	Controls the display of FTP server replies

#### Related reference

“File Transfer Protocol server subcommands” on page 39

These subcommands represent communication between the client and server. This topic includes descriptions for iSeries CL-equivalent subcommands that are unique to iSeries FTP server.

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“File Transfer Protocol server reply status messages” on page 134

This topic contains access common reply codes and what they indicate.

### ACCT (Send Account Information) FTP client subcommand

Some systems require account information to enable certain system functions. The remote system prompts you for such information. To send account information, use the Account (ACCT) FTP client subcommand:

```
ACCT account-information
```

#### account-information

A string that identifies the user’s account. Account information can take the form of a password that the host system uses to grant privileges. This password is not your user password, but rather it is a password on the remote system.

For example, TCP/IP on the IBM Virtual Machine (VM) Operating System might require a password for read and write access to minidisks. Use the ACCT subcommand to supply a password for the minidisk of the current directory. If the remote system is an iSeries, the ACCT subcommand performs no operation.

#### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“APPEND (Append a Local File Member to a Remote File)” on page 60

“DELETE (Delete a File on a Remote System)” on page 64

## **APPEND (Append a Local File Member to a Remote File) FTP client subcommand**

To add a local file member, document, or other file system file to a remote file, use the APPEND FTP client subcommand in this format:

```
Append localfile [remotefile]
```

### **localfile**

The name of the local file member, document, or other iSeries file. The name of the hierarchical file system (HFS) file added to a directory on the remote system.

### **remotefile**

The file on the remote system. If you do not enter a remote file, the FTP client creates a default name.

If the remote file does not exist on the server, the FTP server creates it.

To add a file on the remote system, you must have write privileges to it. You might need to supply the appropriate account information by using the ACCT subcommand (see ACCT (Send Account Information)).

The default file copy mode is stream. You may need to change this by using the MODE subcommand. In the case of fixed-record format in the remote file, the server preserves the file format and record length of the remote file. Records from the local file member shorten or include blanks when necessary.

### **Related reference**

“NAMEFMT (Select File Naming Format)” on page 73

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“ACCT (Send Account Information)” on page 59

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“Naming files for transfer” on page 142

Use the *localfile* and *remotefile* parameters to transfer data.

## **ASCII (Change File Type to ASCII) FTP client subcommand**

```
AScii
```

There are two basic file types you can use when transferring files with FTP: ASCII and BINARY. ASCII files are plain text files. They can have extensions like .txt or have no extension at all. BINARY files are programs or other non-text files saved in the file format of the application that created them or archived or compressed file formats.

Use the ASCII transfer type when transferring text files to or from an ASCII system that does not support EBCDIC representation. ASCII is the default transfer type. The server does not associate a vertical format control to the file. ASCII only supports the default format NON PRINT.

### **Related concepts**

“Data transfer methods” on page 124

Before you begin to transfer files, you must choose the appropriate file transfer type.

**Related reference**

“BINARY (Set Transfer Type to Image)”

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

**Related information**

“Scenario: Transfer a file from a remote host” on page 1

To transfer a file to a test server, use basic File Transfer Protocol (FTP) to send the file to the remote host.

## **BINARY (Set Transfer Type to Image) FTP client subcommand**

Binary
--------

There are two basic file types you can use when transferring files with FTP: ASCII and BINARY. ASCII files are plain text files. They can have extensions like .txt or have no extension at all. BINARY files are programs or other non-text files saved in the file format of the application that created them or archived or compressed file formats.

If you are transferring binary data to an existing iSeries file, the record length is the record length of the existing iSeries file. For example the existing file size should accommodate the new data. If the file does not exist on an iSeries server, FTP chooses a record length for you.

Certain files, such as save files, require binary image transfer. If TYPE is not binary when attempting to transfer such files, you receive a message that tells you to use binary.

**Related concepts**

“Data transfer methods” on page 124

Before you begin to transfer files, you must choose the appropriate file transfer type.

**Related reference**

“ASCII (Change File Type to ASCII)” on page 60

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

**Related information**

“Scenario: Transfer a file from a remote host” on page 1

To transfer a file to a test server, use basic File Transfer Protocol (FTP) to send the file to the remote host.

## **CD (Change Working Directory or Library) FTP client subcommand**

Use the Change Directory (CD) FTP client subcommand to change the working directory, library, or file group on the remote system:

CD <i>directory</i>
---------------------

## directory

The name of a file directory, library, or other system-dependent file-group designator on the remote system.

If the remote system is an iSeries, this subcommand changes the current library or directory. To find out what directories are on the remote system, use the Directory (DIR) subcommand to get a listing.

Use the DIR subcommand with caution.

**Note:** When using the subcommand CD (or LCD) to change from one iSeries file system to another, you must specify the root directory of the file system that contains the new current directory.

### Related reference

“DIR (List Directory Entries, Libraries, or Files)” on page 64

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“LS (List Remote File Names)” on page 69

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

## CLOSE (End an FTP Session with the Remote System) FTP client subcommand

To end your session with the remote system and keep FTP active on your local iSeries, use the CLOSE FTP client subcommand:

```
CLose
```

The CLOSE subcommand allows you to remain in the FTP environment to open another FTP session on another system. Use the OPEN subcommand to establish a new connection with the same remote system or another remote system. Use the QUIT subcommand to end FTP service and return to the iSeries environment from which FTP was started.

### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## DEBUG (Create Client Trace and Control Display of Server Subcommands Sent to Remote System)

**Note:** Only use the FTP client trace to report software problems to IBM. System performance might be affected by this function.

You can use this capability in OS/400 V4R4 and later of the operating system.

### FTP client subcommand

To produce an FTP client trace or display, use the DEBUG FTP client subcommand. The DEBUG subcommand toggles the debugging mode. If the client specifies an optional debug-value, it will use it to

set the debugging level. When debugging is on, the client displays with the string '>>>'. You must set the debug-value to 100 to produce an FTP client trace.

```
DEBUg [debug value]
```

### debug value

If the debug-value is 0, debugging is off. If the debug-value is a positive integer, debugging is on. If you don't specify a value, the debug value toggles from zero to one or from a positive integer to zero.

- 100** Initiate an FTP client trace. The client continues running the trace until the DEBUG is off or until the server ends the FTP client. When the server ends the trace, there might be a significant delay while it formats the trace data.

To initiate a trace immediately when the FTP client starts, you need to create the QTMFTPD100 data area in the QTEMP library by using this command:

```
CRTDTAARA DTAARA(QTEMP/QTMFTPD100) TYPE(*LGL) AUT(*USE)
```

If the QTMFTPD100 data area exists, then it will set the debug value to 100 and start an FTP client trace. The purpose of this capability is to enable the FTP client debug traces in those situations when an FTP client trace *cannot* start with the DEBUG 100 subcommand.

#### Related reference

"File Transfer Protocol client syntax conventions" on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## DEBUG (Change Client Time-Out Limit Values) FTP client subcommand

To change the client time-out limits when the default time-out values are not long enough for a data transfer to complete successfully, use the DEBUG subcommand. You should only need to change these values in situations where network traffic or other conditions cause transfer times to become quite large.

To change the FTP client time-out values, use the DEBUG FTP client subcommand:

```
DEBUg T1 | T2 [ value ]
```

- T1** Change or display the FTP client time-out limit for reading server replies. If the FTP client does not receive an expected server reply within this time limit, the client will close the control connection to the server.
- T2** Change or display the FTP client time-out limit for transferring data. If the FTP client does not receive an expected data connection response within this time limit, the client will close the data connection to the server.
- value** The time-out limit in seconds. This value must be a positive number greater than zero. When you omit this value, the client displays the current value of the time-out limit.

For example:

```
DEBUG T1 900
```

This value sets the client time-out value for server replies to 900 seconds.

#### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

## **DELETE (Delete a File on a Remote System) FTP client subcommand**

Use the DELETE subcommand to delete a file or database file member on a remote system. The DELETE FTP client subcommand format is:

```
DELEte remotefile
```

### **remotefile**

The file you want to delete on the remote system.

The remote system might prompt you for authorization to delete a file. Use the ACCT (Send Account Information) subcommand to respond to that request.

#### **Related reference**

“NAMEFMT (Select File Naming Format)” on page 73

“ACCT (Send Account Information)” on page 59

“MDELETE (Delete Multiple Files on a Remote System)” on page 70

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

## **DIR (List Directory Entries, Libraries, or Files) FTP client subcommand**

The DIR FTP client subcommand displays libraries and their contents or the remote system’s list of directories and directory entries. Use the Directory (DIR) subcommand in this format:

```
DIr [ name] [(Disk]
```

**name** The name of the directory or library. The default is the entire current directory or library. To make a library or directory current, use the Change Working Directory (CD) subcommand. How you specify a set of remote files depends on the system. Most systems allow a generic asterisk, \*. If the remote system is an iSeries, for example: DIR MYLIB/MYFILE. \* produces a list of all members of MYFILE in library MYLIB.

There are two possible file name formats you can use. The example shown here uses NAMEFMT 0. For information about FTP file naming, see “NAMEFMT (Select File Naming Format)” on page 73.

**(Disk** Stores the results of the DIR subcommand in the file \* CURLIB/DIROUTPUT.DIROUTPUT, instead of showing the results on the display.

If the remote system is an iSeries, the information includes:

- For database files, the \*FILE objects, and members.
- For hierarchical file system (HFS) files:
  - All document library services (QDLS) folders and their contents, which can be other folders or documents.
  - All optical volumes (QOPT) and their contents, which can be directories or files.

Use the DIR subcommand with caution. If you enter the DIR subcommand without any parameters, the server produces a listing of all the current directory files. This might be a much longer list than you want.

To get a list of the file names in a directory, use the List (LS) subcommand.

**Related reference**

“CD (Change Working Directory or Library)” on page 61

“NAMEFMT (Select File Naming Format)” on page 73

“LS (List Remote File Names)” on page 69

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **EBCDIC (Change File Type to EBCDIC) FTP client subcommand**

EbCdIc
--------

The EBCDIC transfer type is useful when transferring files to or from another EBCDIC system. This is due to the fact that it avoids the need to convert between ASCII and EBCDIC on both systems.

**Related concepts**

“Data transfer methods” on page 124

Before you begin to transfer files, you must choose the appropriate file transfer type.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **GET (Copy a File from a Remote System to the Local System) FTP client subcommand**

Get <i>remotefile</i> [ <i>localfile</i> ] [(Replace)]
---

**remotefile**

The file that you want to retrieve from the remote system.

**localfile**

The local file member, document, or other file you want to create. If you do not specify a local file name, the FTP client provides a default name. For information about the default names, see Default file names for client transfer subcommands.

## ( Replace

Writes over the localfile if it already exists. The server does not overwrite the localfile unless you specify (Replace.

The file system in which the file resides determines which file name format you use with the GET subcommand.

- If the file does not reside in the library file system (QSYS.LIB), you must use the GET subcommand in name format (NAMEFMT) 1:

```
GET /QDLS/QIWSOS2/PCSMENU.EXE
```

- If the file resides in the library file system, use the GET subcommand and the name format (NAMEFMT) set to 0:

```
GET YOURLIB/YOURFILE.YOURMBR (REPLACE
```

Assuming that the remote server is an iSeries, this command gets the YOURMBR of YOURFILE in YOURLIB and places it in YOURMBR of YOURFILE in your current directory on your local system. For more information about defining the current directory, see “LCD (Change Working Library or Directory on Local System)” on page 67.

**Note:** If the remote file name requires apostrophes as part of the file name, then enclose the file name within two more sets of apostrophes. The following example gets 'MEMBER.ONE' from the remote host. See “Enclose subcommand parameters” on page 139 for more information.

```
GET LIBRARY/FILE.MEMBER 'MEMBER.ONE'
```

### Related reference

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“LCD (Change Working Library or Directory on Local System)” on page 67

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“MGET (Copy Multiple Files from a Remote System to the Local System)” on page 71

“PUT (Copy a File Member from the Local System to a File on a Remote System)” on page 75

“MPUT (Send Multiple File Members from the Local System to a Remote System)” on page 73

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

## HELP (Getting Help for FTP Subcommands)

The HELP subcommand provides information about the FTP subcommands that the local system and the remote system uses.

### Help for FTP client subcommands

To get information about FTP subcommands used by the local system, use the HELP subcommand in this format:

```
Help [* | ALL | subcommand ]
```

#### \* or ALL

Displays a list of the FTP client subcommands.



### **subcommand**

Provides detailed help for the specified client subcommand. For example, HELP GET tells you how to transfer a file from a remote system to your local system. You may abbreviate the subcommand to a meaningful prefix.

If you use the HELP subcommand without a parameter, you see a list of subcommands and a general description of the help information available. Context-sensitive help is available by positioning the cursor over a command on the help display and then pressing the Enter key.

To get the list of local subcommands on an iSeries server, enter:

```
HELP
```

Help information can be obtained with the ? subcommand.

## **Help for FTP server subcommands**

To obtain help for FTP subcommands on the remote system, use the HELP subcommand in this format:

```
Help SERVER [subcommand]
```

### **SERVER**

Gives the help the remote system offers for FTP server subcommands. This is similar to using QUOTE with the HELP parameter. QUOTE HELP lists the FTP subcommands supported by the remote system.

### **subcommand**

The name of the server subcommand that you want the information. For example, HELP SERVER STOR will request the server to provide help on the STOR subcommand.

**Note:** RHELP is a synonym for HELP SERVER. For example, HELP SERVER SITE and RHELP SITE are equivalent.

#### **Related reference**

“QUOTE (Send a Subcommand to an FTP Server)” on page 77

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **LCD (Change Working Library or Directory on Local System) FTP client subcommand**

```
LCd pathname
```

### **pathname**

The name of a library, folder, or directory on the local system.

#### **Notes:**

1. The LCD subcommand does not change the current library entry of the library list.
2. When using the subcommand CD (or LCD) to change from one file system to another file system, you must specify the “root” directory, for example, /QDLS or /QOPT.

#### **Related reference**

“GET (Copy a File from a Remote System to the Local System)” on page 65

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

## **LOCSITE (Specify Local Site Information) FTP client subcommand**

To specify information that is used by the FTP client to provide services specific to the client system, use the LOCSITE FTP client subcommand:

LOCSITE [ parameters ]

The iSeries FTP client supports these parameters and parameter options for the LOCSITE subcommand:

### **CRTCCSID \*CALC**

New database files created during ASCII file transfers use the related default EBCDIC CCSID of the ASCII file transfer CCSID. This is the default value.

### **CRTCCSID \*USER**

New database files created during ASCII file transfers use the current job CCSID. However, if this CCSID is 65535, the default CCSID determined by the language id in the current job specification.

### **CRTCCSID \*SYSVAL**

New database files created during ASCII file transfers use the CCSID that the QCCSID system value specifies.

### **CRTCCSID [CCSID-number]**

Specify the CCSID you want to use when creating database files on the client during ASCII file transfers. The server validates this value.

### **CRTCCSID**

Display a message that contains the current FTP client CRTCCSID setting.

### **TRIM 0**

Set Trim option to OFF. The server sends trailing blanks of database records.

### **TRIM 1**

Set Trim option to ON. The server does not send trailing blanks of database records when transferring database files that use file structure and stream mode. This is the default.

### **TRIM 2**

Set Trim option so the server does not send trailing blanks of database records for all transfers, including record structure and block mode.

**TRIM** Display a message that contains the current setting of the FTP client TRIM option.

#### **Notes:**

1. Prior to the availability of this subcommand, trailing blanks of QSYS.LIB file system records were always removed before transferring the file to the server system.
2. TRIM settings do not apply to TYPE I (binary) file transfers. Blanks are never trimmed for TYPE I file transfers, regardless of the TRIM setting.

### **DTAPROT C**

Set the data protection variable to C (Clear). This variable is used to set the data protection level

when opening a secure control connection. For more details about setting data protection security, refer to the following subcommands: SECDATA and SECOPEN.

#### **DTAPROT P**

Set the data protection variable to P (Private). This variable is used to set the data protection level when opening a secure control connection.

#### **DTAPROT**

Display a message that contains the current value of the data protection variable.

##### **Related reference**

“SECDATA (Setting Data Security Protection)” on page 79

“SECOPEN (Setting Data Security Protection)” on page 80

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

### **LOCSTAT (Display Local Status Information) FTP client subcommand**

LOCSTat

Displays Local status information, including:

- The current setting of the SENDSITE subcommand
- The current setting of the SENDPORT subcommand
- Remote system name, port number, and logon status
- Data type and transfer mode
- Name format value for both the client and the server
- Setting for the VERBOSE mode
- Setting for the DEBUG mode

##### **Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

### **LS (List Remote File Names) FTP client subcommand**

LS [ *name*] [(Disk)]

**name** The remote directory, file, or library that you want to list. If the remote system is an iSeries server, the server lists the file names and its members. The default is to list the entire current directory, library, or folder. To change the current directory, library, or folder, use the CD subcommand. The remote file specification is system dependent.

**(Disk** Stores the results of the LS subcommand in the file \*CURLIB/LSOUTPUT.LSOUTPUT, instead of showing the results on the display. Each time you specify the (Disk parameter with the same \*CURLIB, the server changes the contents of the LSOUTPUT.LSOUTPUT member file.

**Note:** If the FTP server returns a negative reply code (550), then there will be no LSOUTPUT member. If the FTP server returns a positive reply code (150) without any file names, then an LSOUTPUT member with no records will result.

The LS subcommand lists the file names only. To get a list of complete directory entries with additional information about the files, see “DIR (List Directory Entries, Libraries, or Files)” on page 64.

**Related reference**

“DIR (List Directory Entries, Libraries, or Files)” on page 64

“CD (Change Working Directory or Library)” on page 61

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **LTYPE (Local Type) FTP client subcommand**

To specify the file transfer type or the representation in which the transfer is to take place on the local system, use the LTYPE FTP client subcommand in this format:

```
LType C ccsid#
```

**C** The CCSID type. Code this value as C.

**ccsid#** The CCSID value. Code this value as a CCSID number 1-65533.

**Note:** The LTYPE subcommand is similar to the TYPE subcommand. The LTYPE subcommand changes only the representation type on the client side. The TYPE subcommand changes the representation type on both the client and the server.

**Related reference**

“TYPE (Specify File Transfer Type)” on page 86

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **MDELETE (Delete Multiple Files on a Remote System) FTP client subcommand**

```
MDelete { remotefile [remotefile...]}
```

**remotefile**

The file or files on the server system you want to delete.

**Note:** When the remotefile is an iSeries QSYS.LIB file, then the server deletes all members of the iSeries physical file. The file itself remains.

This is a library file system example in NAMEFMT 0:

```
MDELETE MYLIB/FILE1.MBRA YOURLIB/FILE2.MBRB
```

This deletes member MBRA in file FILE1 in library MYLIB and member MBRB in file FILE2 in library YOURLIB on a remote iSeries server. The same example in NAMEFMT 1:

```
MDELETE /QSYS.LIB/MYLIB.LIB/FILE1.FILE/MBRA.MBR  
/QSYS.LIB/YOURLIB.LIB/FILE2.FILE./MBRB.MBR
```

This is a document library system example in NAMEFMT 1:

```
MDELETE /QDLS/QIWSOS2/PCSMENU.EXE /QDLS/PCSDIR/PCSFILE.EXE
```

This deletes document PCSMENU.EXE in folder QIWSOS2 in the document library services library, and also deletes PCSFILE.EXE in folder PCSDIR in the QDLS library on an iSeries.

You can use an asterisk (\*) to delete the files generically. For example with NAMEFMT 0, if the remote system is an iSeries, type:

```
MDELETE MYLIB/MYFILE.*
```

This example would delete all members of file MYFILE in library MYLIB. Use of the asterisk is only valid at the end of a character string.

#### **Related reference**

“DELETE (Delete a File on a Remote System)” on page 64

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

## **MGET (Copy Multiple Files from a Remote System to the Local System) FTP client subcommand**

### **How MGET transfers files:**

To copy one or more remote files, use the MGET FTP client subcommand in this format. A separate GET subcommand is executed for each remote file you want transferred. The server creates the name of the corresponding local file automatically as determined by the Default Naming rules.

The MGET FTP client subcommand uses the following process to determine where to put files.

- The MGET subcommand always places files in the current library or directory.
- If the user has issued the LCD subcommand, the server uses this library or directory.
- If the user has not issued the LCD subcommand, the server sets the current directory as follows
  - If the user’s job has a current library set, this library is the current directory for FTP.
  - If the user’s job does not have a current library set, the server uses QGPL as the current directory.

MGet {remotefile [remotefile...]}[(Replace)]
---

#### **remotefile**

The file or files you want to retrieve from the remote system.

#### **( Replace**

Overwrites an existing file on your local system. If the file already exists on your local system and you do not use the Replace option, the existing file is not overwritten. The name of the local file where the remotefile is copied is created automatically.

You can use an asterisk (\*) to copy all members in a file to your current library or directory. For example, if the remote system is an iSeries,

- MGET MYLIB/MYFILE. \* copies all the members of file MYFILE in library MYLIB on the remote system to your current library on the local system.
- MGET /QSYS.LIB/MYLIB.LIB/MYFILE.FILE/ \* .MBR would be the NAMEFMT 1 version of this command.

- MGET /QOPT/PICTURES/IMAGES/. \* copies all the files of directory IMAGES from optical volume PICTURES to your current library (or directory) on the local system.
- MGET TESTFILE.A \* copies all members that start with the letter A in file TESTFILE.
- MGET /QDLS/QISSOS2/A \* copies all documents that start with the letter A in folder QISSOS2.

**Related reference**

“GET (Copy a File from a Remote System to the Local System)” on page 65

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“MPUT (Send Multiple File Members from the Local System to a Remote System)” on page 73

“PUT (Copy a File Member from the Local System to a File on a Remote System)” on page 75

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

**MKDIR (Make Directory)  
FTP client subcommand**

MKdir pathname

**pathname**

The name of a file directory, library, or other system-dependent file-group designator on the remote system.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

**MODE (Specify Transmission Mode of Data)  
FTP client subcommand**

MODE [ B | S ]

- B** Specifies block mode. In this mode, the server transmits data as a series of data blocks, preceded by one or more header bytes. If you are transferring data in block mode, the type must be EBCDIC.
- S** Specifies stream mode. In this mode, the server transmits data as a stream of bytes. You can use any representation type with stream mode.

**Notes:**

1. Stream mode is the default transfer mode that is used in FTP. Some systems do not support block mode.
2. If you omit the optional parameter, the client displays the present MODE value.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **MPUT (Send Multiple File Members from the Local System to a Remote System) FTP client subcommand**

To send one or more local files to the remote system, use the MPUT FTP client subcommand. The client executes a separate PUT subcommand for each local file you want transferred. The Default Naming rules create the name of the corresponding remote file.

```
MPut { localfile [localfile...]}
```

### **localfile**

Specify one or more local library file system file members or other FTP supported file system files you want transferred to the remote system. The client automatically generates the name given to the file on the remote system.

**Note:** If the remote file already exists, the contents are replaced by the contents of *localfile* unless Store Unique (SUNIQUE) is on.

For information as to how to specify the file if the remote system is an iSeries, see “NAMEFMT (Select File Naming Format).” This example uses NAMEFMT 0:

```
MPUT MYLIB/FILE1.MBR1 MYLIB/FILE1.MBR2
```

This sends members MBR1 and MBR2 of file FILE1 in library MYLIB to the remote system.

This example uses NAMEFMT 1:

```
MPUT /QDLS/QIWSOS2/PCSMENU.EXE /QDLS/QIWSOS2/PCSMENU2.EXE
```

This sends document PCSMENU.EXE and document PCSMENU2.EXE from folder QIWSOS2 to the remote system.

You can use an asterisk (\*) to send all the members in a file. For example, MPUT MYLIB/MYFILE. \* transfers all the members of file MYFILE in library MYLIB.

### **Related reference**

“GET (Copy a File from a Remote System to the Local System)” on page 65

“SUNIQUE (Control Overwriting of Files)” on page 84

“MGET (Copy Multiple Files from a Remote System to the Local System)” on page 71

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“PUT (Copy a File Member from the Local System to a File on a Remote System)” on page 75

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the *localfile* and *remotefile* parameters to transfer data.

## **NAMEFMT (Select File Naming Format) FTP client subcommand**

To select which file name format to use on the local system and the remote system (if it is an iSeries), use the NAMEFMT FTP client subcommand:

NAMEfmt [ 0 | 1 ]

- 0 A name format only for library file system database files. The general format is:  
[libname/]filename[.mbrname]
- 1 A name format for all file systems that FTP supports, including the library file system. You must set the name format to '1' to work with all iSeries file systems.

Library file system files in this name format are:

[/QSYS.LIB/][libname.LIB/]filename.FILE[/mbrname.MBR]

For save files, you can also use the format:

/QSYS.LIB/libname.LIB/filename.SAVF

Files in the document library services file system are in this format:

[/QDLS/][{foldername[.ext]/}]filename[.ext]

For optical, the format is:

/QOPT/volname/dirname/filename.ext

**Notes:**

- 1. You can set the name format to 0 only when the working directory is a database library.
- 2. If you specify the NAMEFMT subcommand without a parameter, the client displays the current name format.

**Related reference**

“APPEND (Append a Local File Member to a Remote File)” on page 60

“DELETE (Delete a File on a Remote System)” on page 64

“DIR (List Directory Entries, Libraries, or Files)” on page 64

“File systems and naming conventions” on page 133

The File Transfer Protocol (FTP) server arranges the information units of a file system in a multiple-level tree-like structure.

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

**NULLFLDS (Allow Transfer of Files with NULL Fields)  
FTP client subcommand**

Use this command to select whether or not to allow transfer of a database files that contain NULL field values on the local system, and the remote system if it is an iSeries.

NULLflds [ 0 | 1 ]

When you enter a parameter the valid values are:

- 0 Do not allow transfer of database files that contain NULL fields. This is the default.
- 1 Allow transfer of database files that contain NULL fields.

**Notes:**



1. Transfer of files that contain NULL fields requires both the client and server to have this setting enabled. The target file must exist prior to the file transfer. Also, the target file must have the same file definition as the source file.
2. Results are not predictable if you transfer a file that contains NULL fields a server that is not an iSeries, or if the transfer type results in codepage conversion of the data.
3. If you specify the NULLFLDS subcommand without a parameter, the client displays the current setting.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **OPEN (Connect to FTP Server on a Remote System) FTP client subcommand**

Open *systemname* [*portnumber*]

**systemname**

The name or Internet address of the remote system.

**portnumber**

The port number to use for this session until the server closes the connection. This is optional. If you do not specify a port number, the server chooses one.

After you have opened a connection to a remote system, you cannot connect to another system until you close the current session.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **PASS (Send Your Password) FTP client subcommand**

PAss *password*

**password**

A string that specifies your password.

The OPEN and USER subcommands must precede this subcommand. For some systems, this completes your identification for access control. This subcommand is not necessary when the server requests you to type a password when connecting or logging on to the server.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **PUT (Copy a File Member from the Local System to a File on a Remote System) FTP client subcommand**

```
PUT localfile [remotefile]
```

**localfile**

The name of a local library system file member, save file, document, or other file.

**remotefile**

The name of the delivered file on the remote system. If you do not specify the remote file name, the FTP server provides a default name. If a remote file with the same name already exists, then the server replaces the contents with the contents of the local file unless Store Unique (SUNIQUE) is on.

To send a file to the remote system, you must have a defined current working directory with write privileges.

This example uses the PUT subcommand to transfer a file member:

```
PUT MYLIB/MYFILE.MYMBR (NAMEFMT = 0)
```

This sends member MYMBR of file MYFILE in library MYLIB to the remote system.

This example sends the document PCSMENU.EXE of folder QIWSOS2 in the document library services file system to the remote system.

```
PUT /QDLS/QIWSOS2/PCSMENU.EXE (NAMEFMT = 1)
```

**Note:** If the remote file name requires apostrophes as part of the file name, then you must enclose the file name within two more sets of apostrophes. The following example sends 'MEMBER.ONE' as the file name to the remote host.

```
PUT LIBRARY/FILE.MEMBER 'MEMBER.ONE'
```

**Related tasks**

“Enclose subcommand parameters” on page 139

You can use either a single quotation mark (') or quotation marks (") to enclose parameters from this topic.

**Related reference**

“GET (Copy a File from a Remote System to the Local System)” on page 65

“MPUT (Send Multiple File Members from the Local System to a Remote System)” on page 73

“File names for client-transfer subcommands” on page 140

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“SUNIQUE (Control Overwriting of Files)” on page 84

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“MGET (Copy Multiple Files from a Remote System to the Local System)” on page 71

“Naming files for transfer” on page 142

Use the localfile and remotefile parameters to transfer data.

**PWD (Display Current Directory, Folder, or Library)  
FTP client subcommand**

To display the current directory or library of the remote system, use the PWD FTP client subcommand:

PWd

If the remote server is an iSeries, the server displays your current library or file system directory on the remote system. Also, the server displays the working directory in quotation marks. To change the current library or directory of the remote system, use the Change Working Directory (CD) subcommand.

#### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## QUOTE (Send a Subcommand to an FTP Server) FTP client subcommand

QUOTE *string*

**string** The server subcommand you want sent to and interpreted by the remote FTP server. The FTP server sends the string verbatim to the remote FTP server.

#### Notes:

1. The client requires the QUOTE subcommand to run the special iSeries FTP server subcommand RCMD (Send a CL Command to an FTP Server System). For example, to write the server job log to a spooled file, enter this:  
QUOTE RCMD DSPJOBLOG  
You can use WRKSPLF to access the job log. Note that you will need to specify the user profile of the user who logged in to the FTP server if the WRKSPLF is run from a different user profile.
2. iSeries FTP server limits the string to 1000 characters.
3. For the QUOTE subcommand, whatever you enter passes on to the server. For example, if you enter:

```
QUOTE CWD 'SYS1'
```

The server receives

```
CWD 'SYS1'
```

You can get help information from the server by typing this:

```
QUOTE HELP
```

The server sends the HELP subcommand to the remote host, which returns a display of all subcommands it supports. The information displayed varies depending on the type of remote host.

It should be noted that server subcommands entered with the QUOTE subcommand only affect the server, but similar client subcommands might affect both the client and the server. For example, the REIN client subcommand sends the server a REIN server subcommand plus reinitializes certain client state variables. QUOTE REIN sends only REIN to the server, but does not change any client state variables.

**Note:** Be careful when using the QUOTE subcommand to directly enter server subcommands so that unintended results do not occur. Typically, use the QUOTE subcommand for special situations that cannot use other client subcommands. An example of this is when one wants to use one of the special iSeries server subcommands like CRTL.

#### Related concepts

“Server timeout considerations” on page 29  
You can keep your connection from timing out.

**Related reference**

“HELP (Getting Help for FTP Subcommands)” on page 66  
“RCMD (Send a CL Command to an FTP Server System)” on page 50  
WRKSPLF

“File Transfer Protocol client syntax conventions” on page 137  
The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“File systems and naming conventions” on page 133  
The File Transfer Protocol (FTP) server arranges the information units of a file system in a multiple-level tree-like structure.

## **REINITIALIZE (Reinitialize Session between Systems) FTP client subcommand**

REInitialize
--------------

If the server supports the REINITIALIZE subcommand, the USER session with the server is ended. The server is in the same state as when the connection was established, and the user needs to log on again to continue.

Any file transfers already in progress can complete before the USER session ends.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137  
The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **RENAME (Rename a File on a Remote System) FTP client subcommand**

REname <i>originalname newname</i>
------------------------------------

**originalname**

The present name of the remote file.

**newname**

The new name of the remote file. If the file specified by *newname* already exists, the new file replaces it.

This example renames the file SPORTSCAR.BMP in directory IMAGES on optical volume PICTURES to CAR.BMP:

```
REN /QOPT/PICTURES/IMAGES/SPORTSCAR.BMP  
/QOPT/PICTURES/IMAGES/CAR.BMP
```

**Note:** On an iSeries server, you cannot rename a file to a different file system.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137  
The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## RESET (Reset) FTP client subcommand

To clear the server reply queue, use the RESET FTP client subcommand:

```
REset
```

This subcommand resynchronizes the sequencing of the server subcommands and replies with the remote FTP server. Resynchronization might be necessary following a violation of the FTP protocol by the remote server.

### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## RMDIR (Remove Directory) FTP client subcommand

```
RMdir pathname
```

### pathname

The name of a file directory, library, or other system-dependent file-group designator on the remote system. For hierarchical file system (HFS) directories, you can only delete empty directories. The server deletes Libraries unconditionally.

### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## SECData (Setting Data Security Protection) FTP client subcommand

To specify the protection level to be used for the data connection when there is a secure control connection already established with the remote system, use the SECData subcommand as follows:

```
SECData [ C | P ]
```

**Note:** SData is a synonym for this subcommand.

- C** Data channel protection level is set to 'clear'. This connection is **not** secure. This might be used for pre-encrypted data or non-sensitive data.
  - P** The data channel protection level is set to 'private'. This connection is secure. A Transport Layer Security (TLS) negotiation between the client and the server must take place before any data is transmitted over the connection.
1. When no parameter is specified, SECData displays the present value used for setting data security protection.
  2. The data protection level is initially set to the value specified by the DTAPROT parameter of the STRTCPFTP CL command when a secure control connection is established with an FTP server.
  3. A secure control connection is required to use the SECData subcommand.

4. A PROT server subcommand is issued to the server each time the SECDATA subcommand successfully sets the data protection level.
5. The SECDATA subcommand sends a PBSZ and a PROT subcommand to the server when setting the data protection level. Also, the SECDATA subcommand sets a client variable for each successful PROT subcommand. This variable represents the last data protection level (C or P) accepted by the server. This variable is used to set the data protection level when the SECOpen subcommand opens a secure control connection. This variable can be changed using the LOCSITE DTAPROT option.
6. The parameters 'C' and 'P' for the SECDATA subcommand are the same as used by the PROT server subcommand.

**Related concepts**

“Secure the FTP client with Transport Layer Security or Secure Socket Layer” on page 20  
 You can use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) connections to encrypt data transferred over File Transfer Protocol (FTP) control and data connections.

**Related reference**

“LOCSITE (Specify Local Site Information)” on page 68  
 “File Transfer Protocol client syntax conventions” on page 137  
 The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

**SECOpen (Setting Data Security Protection)  
 FTP client subcommand**

The SECOpen FTP client subcommand opens a secure control connection to an FTP server using the specified security option. The syntax of this subcommand is:

```
SECOpen systemname [portnumber] [security_option ]
```

**Note:** SOpen is a synonym for SECOPEN.

**systemname**

Enter the name or Internet address of the remote system.

**portnumber**

Enter the port number for this connection.

**Notes:**

- If this parameter is omitted and (SSL is specified, the port number 21 will be used.
- If this parameter is omitted and (IMPLICIT is specified, then port number 990 is used.
- If both the port number and the security\_option are omitted, then port number 21 and (SSL are assumed.

**security\_option**

Specify the type of security to be used.

**(SSL)** Uses a secure SSL connection to the FTP server. The AUTH (Authorization) server subcommand is used when making the connection.

**(IMPLICIT)**

Uses an "implicit" SSL or TLS secure connection to the FTP server. An "implicit" SSL connection is made without sending the AUTH, PBSZ, and PROT sever subcommands to the server. In this case, the server must be configured to expect an SSL/TLS connection negotiation to occur for the specified port number.

For the "implicit" SSL case, the server will act as if the client had sent these subcommands with the parameters shown below:

- AUTH SSL
- PBSZ 0
- PROT P

**Note:** If the security\_options parameter is not specified, then (SSL will be assumed. When the port number used is 990, then (IMPLICIT is assumed.

#### **Related concepts**

"Secure the FTP client with Transport Layer Security or Secure Socket Layer" on page 20

You can use Transport Layer Security (TLS) or Secure Sockets Layer (SSL) connections to encrypt data transferred over File Transfer Protocol (FTP) control and data connections.

#### **Related reference**

"LOCSITE (Specify Local Site Information)" on page 68

"File Transfer Protocol client syntax conventions" on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **SENDPASV (Specify Whether to send a PASV Subcommand) FTP client subcommand**

To specify whether or not to send a PASV subcommand to the FTP server when doing a data transfer or issuing the DIR and LS subcommands, enter the SENDPASV FTP client subcommand:

```
SENDPASV [ 0 | 1 ]
```

If there is no parameter SENDPASV works like a toggle switch. The SENDPASV value toggles from 1 (ON) to 0 (OFF) or from 0 to 1.

When there is a parameter, the valid values are:

- 0** Do not send a PASV subcommand.
- 1** Send a PASV subcommand. This is the default.

iSeries default (on) is to send the PASV subcommand. When SENDPASV is off, then the server does not send the PASV subcommand.

#### **Notes:**

1. This subcommand supports RFC 1579, "Firewall-Friendly FTP." Use of the PASV subcommand to establish a data connection is a better method when a data transfer must go through a firewall. In some scenarios, a data transfer through a firewall might not be possible without use of PASV.
2. Some FTP servers might not support the PASV subcommand. When this is the situation and SENDPASV is ON, then the FTP client will display a message that indicates that the server does not support PASV. The system will attempt to establish the data connection without sending the PASV subcommand.
3. When SENDPASV is OFF or disabled, then the server sends the PORT subcommand when SENDPORT is ON.
4. FTP servers that do not support PASV are not compliant with RFC 1123.

#### **Restriction:**

When connected to an FTP server through a SOCKS server, you can only use the SENDPASV subcommand before you issue a data transfer subcommand or a list directory subcommand. If you use SENDPASV after one of these subcommands, then the client cannot be able to establish a data connection to the FTP server.

After the client has issued a data transfer or list directory subcommand, then close the connection to the FTP server through a SOCKS server before you issue SENDPASV again.

You can use the SENDPASV subcommand when the FTP client is disconnected from an FTP server.

#### **Related reference**

“SENDPORT (Specify Whether to Sends a PORT Subcommand)”

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

#### **Related information**

RFC index search engine

RFC editor

## **SENDPORT (Specify Whether to Sends a PORT Subcommand) FTP client subcommand**

To specify whether to send a PORT subcommand to the FTP server when doing a data transfer or when issuing the DIR and LS subcommands. Enter the SENDPORT FTP client subcommand:

```
SENDPOrt [ 0 | 1 ]
```

If there is no parameter, SENDPORT works like a toggle switch. The SENDPORT value changes from 1 (ON) to 0 (OFF) or from 0 to 1.

When there is a parameter, the valid values are:

- 0** Do not send a PORT subcommand.
- 1** Send a PORT subcommand. This is the default.

#### **Notes:**

1. Use SENDPORT only when you cannot establish a connection to the server without it. The indiscriminate use of SENDPORT might result in errors.
2. You might find it useful to not send the PORT subcommand to those systems that ignore PORT subcommands because they indicate that they have accepted the command.
3. The server does not send the PORT subcommand when the SENDPASV option is ON.

#### **Related reference**

“SENDPASV (Specify Whether to send a PASV Subcommand)” on page 81

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:



## SENDSITE (Specify Whether to Send a SITE Subcommand) FTP client subcommand

To specify whether a SITE subcommand with record format information is automatically sent when doing a PUT or an MPUT operation, enter the SENDSITE FTP client subcommand in this format:

```
SENDSite [ 0 | 1 ]
```

If there is no parameter, SENDSITE works like a toggle switch. The SENDSITE value changes from 0 (OFF) to 1 (ON) or from 1 to 0.

When there is a parameter, the valid values are:

- 0 Do not send a SITE subcommand. This is the default.
- 1 Send a SITE subcommand (containing record format information) before sending PUT and MPUT subcommands. Use this setting when transferring files to an IBM Virtual Machine server that uses the record format information that sends with the SITE subcommand.

### Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“SITE (Send Information Used by a Remote System)”

## SITE (Send Information Used by a Remote System) FTP client subcommand

To send information that is used by the remote system to provide services specific to the remote system, use the SITE FTP client subcommand in this format:

```
Site [parameters]
```

### parameters

Dependent on the remote system.

To find the nature of these parameters and their syntax specifications, issue the HELP SERVER SITE subcommand. Some FTP servers do not support the SITE subcommand.

**Note:** The SITE subcommand is used by the PUT and MPUT subcommands to indicate the format and length of the records. By default, the PUT subcommand sends a SITE subcommand automatically. The NAMEFMT subcommand uses the SITE subcommand to indicate to the server whether names are in NAMEFMT 0 or NAMEFMT 1.

### Related reference

“SENDSITE (Specify Whether to Send a SITE Subcommand)”

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## STATUS (Retrieve Status Information from a Remote System) FTP client subcommand

STatus [*name*]

**name** The name of the remote directory or file for which you request the status information. It is not a required parameter.

**Note:** The iSeries FTP server application does not support this name parameter.

If there is no parameter, the server returns general status information about the FTP server process. This includes current values of all transfer parameters and the status of connections. The status information that is returned depends on the specific server implementation.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **STRUCT (Specify File Structure) FTP client subcommand**

To specify the structure of the data sent for a file, use the STRUCT FTP client subcommand in this format:

STRuct [F | R]

**F** A file structure. The structure of a file is a continuous sequence of data bytes.

**R** A record structure. The file transfers as a sequence of sequential records.

The structure of a file affects the transfer mode and the interpretation and storage of a file.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **SUNIQUE (Control Overwriting of Files) FTP server subcommand**

SUNIQUE is a separate command that has to be issued before the PUT or MPUT. SUNIQUE sets a “mode” (in the same manner as NAMEFMT, LISTFMT, and so on.), so that every PUT/MPUT after it uses the setting entered on the SUNIQUE subcommand. For example:

```
FTP> SUNIQUE 1
FTP> MPUT *.FILES
```

If there is no parameter, SUNIQUE acts like a toggle switch. The SUNIQUE value changes from 0 (OFF) to 1 (ON) or from 1 to 0.

When there is a parameter, the valid values are:

**0** Overwrite the file if it exists. This is the default.

**1** Create a new file with a unique name on the remote system instead of overwriting an existing file. The FTP server on the remote system sends the name of the created file back to the user.

**Note:** If the remote system is an iSeries, the server forms File.Mbr names by adding numbers to the end of the of the *localfile* that you specified in the PUT or MPUT subcommand. Thus, if the name *NEWFILE.NEWMBR* already exists on the remote system, the remote iSeries server creates *NEWFILE.NEWMBR1* and writes the data to it.

File names for other file systems, like HFS, work in a similar way. If the name already exists, a new file is created that consists of the specified file name and a number suffix. Thus, if the name *xfsname* already exists on the remote system, the remote iSeries creates *xfsname1*.

#### **Related reference**

“MPUT (Send Multiple File Members from the Local System to a Remote System)” on page 73

“PUT (Copy a File Member from the Local System to a File on a Remote System)” on page 75

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **SYSCMD (Pass an iSeries CL Command to Your Local iSeries) FTP client subcommand**

To run a control language (CL) command on your local iSeries without leaving the FTP environment, use the SYSCMD FTP client subcommand in this format:

*SYSCmd commandline*

#### **commandline**

An iSeries CL command. You can precede the command name with a ? to get the prompt for the CL command. For example, if you enter:

```
SYSCMD ? SNDBRKMSG
```

you get the display for the Send Break Message (SNDBRKMSG) command.

If you want to see low level messages that result from your CL command, or if you want to enter multiple CL commands before returning to the FTP environment, use the iSeries CALL QCMD command.

For example, to get to an iSeries Command Entry display, enter this:

```
SYSCMD CALL QCMD
```

From the Command Entry display, you can then call your application programs or enter CL commands. At the completion of your application program or the CL command, you return to the Command Entry display. From there you can display messages, start additional work on the system, or press F3 (Exit) or F12 (Cancel) to return to FTP.

You can enter iSeries CL commands when you press F21 (CL command line) from the main FTP display. The server does not allow the F21 key when an exit program it is an addition to the FTP Client Request Validation exit point.

#### **Notes:**

1. Most server systems have a time-out period that ends the session if no activity occurs within a specific time period. If the command runs for longer than the time-out period, the server ends the connection with the client.
2. The iSeries server supports the exclamation mark (!) as a synonym for the SYSCMD subcommand.
3. The SYSCMD subcommand passes to the iSeries as a CL command exactly what the user enters.

## Related reference

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## TYPE (Specify File Transfer Type) FTP client subcommand

To specify the file-transfer type, or the representation in which the transfer is to take place, use the TYPE FTP client subcommand in this format:

```
TYpe [ A
      B [ 1 | 2 | 3 [A|R] | 4 [A|R] | 5 | 6 | 7]
      C ccsid#
      E
      F [ 1 ]
      I ]
```

**A** Specifies the transfer type as the default (ASCII) transfer type. This has the same effect as the ASCII subcommand. The server does not associate any vertical format control with the file. It only supports the default format NON PRINT for ASCII. Use the ASCII transfer type or the transfer of text files, except when both systems use the EBCDIC type.

The default CCSID for TYPE A (ASCII) is the CCSID that is specified on the CCSID parameter of the STRTCPFTP command or FTP subcommand.

**B** Shift JIS Kanji (CCSID 932)

**B 1** Shift JIS Kanji (CCSID 932)

**B 2** Extended UNIX Code Kanji (CCSID 5050)

**B 3** JIS 1983 using ASCII shift-in escape sequence (CCSID 5054)

**B 3 A** JIS 1983 using ASCII shift-in escape sequence (CCSID 5054)

**B 3 R** JIS 1983 using JISROMAN shift-in escape sequence (CCSID 5052)

**B 4** JIS 1978 using ASCII shift-in escape sequence (CCSID 5055)

**B 4 A** JIS 1978 using ASCII shift-in escape sequence (CCSID 5055)

**B 4 R** JIS 1978 using JISROMAN shift-in escape sequence (CCSID 5053)

**B 5** Hangeul (CCSID 934)

**B 6** Korean Standard Code KSC-5601, 1989 version (CCSID 949)

**B 7** Traditional Chinese (5550) (CCSID 938)

### C ccsid#

Specifies the transfer type to any CCSID (coded character set identifier) that is installed on the system. The CCSID number must follow C.

**E** Specifies the transfer type as EBCDIC. This has the same effect as the EBCDIC subcommand. The server does not associate any vertical format control with the file. It only supports the default format NON PRINT for EBCDIC. Use the EBCDIC transfer type for the efficient transfer between systems that use EBCDIC as their internal character representation.

**F** IBM EBCDIC Kanji (CCSID 5035)

**F 1** IBM EBCDIC Kanji (CCSID 5035)

**I** Specifies the transfer type as image. This has the same effect as the BINARY subcommand. With

the image transfer type, data is a string of bits, packed into 8-bit bytes. The image transfer type is an efficient at storing and retrieving files and for transferring binary data such as object code. Data is transferred as is; there is no conversion.

If there are no parameters, the server displays the present setting for the TYPE subcommand.

**Related reference**

“LTYPE (Local Type)” on page 70

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

“Specify mapping tables” on page 130

For File Transfer Protocol (FTP) client, the ASCII mapping tables are specified in the FTP command.

For FTP server this is done in the Change FTP Attributes (CHGFTP) command.

## **USER (Send Your User ID to the Remote System) FTP client subcommand**

User *userid* [*password*]

**userid** Your logon name on the remote system.

**password**

Your password on the remote system. Specifying your password is optional. If you do not supply your password when calling the USER subcommand, you receive a prompt to do so if the remote system requires a logon password.

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **VERBOSE (Control of Text Display of Error Reply Messages) FTP client subcommand**

To control the display of FTP server replies, use the VERBOSE FTP client subcommand. The VERBOSE subcommand toggles the verbose switch on and off. When verbose is on, all server replies, including their reply codes, are displayed. When set to off, certain server replies and reply codes are discarded and not displayed.

Verbose

**Related reference**

“File Transfer Protocol client syntax conventions” on page 137

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:

## **File Transfer Protocol exit programs**

You can use File Transfer Protocol (FTP) exit programs to secure FTP. The FTP server communicates with each exit program through a specific exit point. This topic includes parameter descriptions and code examples.

The FTP client and the FTP server communicate with each exit program through a specific exit point. Parameters are passed between the server and the exit program. The format of the exchanged information is specified by an exit point format.

FTP uses the following exit points. Refer to these topics for more information, including parameter descriptions and code examples:

- Request validation exit point: client and server
- Server logon exit point

To allow the exit programs to work properly, you must install and register your exit point programs. If your programs are no longer needed, you must properly remove the exit point programs to prevent their future functioning.

## TCP/IP exit points and exit point formats

The following table provides information about exit points for various TCP/IP applications and their related exit point formats.

TCP/IP Exit Points	Application	VLRQ0100	TCPL0100	TCPL0200	TCPL0300
QIBM_QTMF_CLIENT_REQ	FTP	X			
QIBM_QTMF_SERVER_REQ	FTP	X			
QIBM_QTMF_SVR_LOGON <sup>1</sup>	FTP		X	X	X <sup>2</sup>
QIBM_QTMX_SERVER_REQ	REXEC	X			
QIBM_QTMX_SVR_LOGON <sup>1</sup>	REXEC		X		X <sup>2</sup>
QIBM_QTOD_SERVER_REQ	TFTP	X			

<sup>1</sup> - An exit point can have more than one format, but an exit program can only be registered for one of the exit point formats. Examine each of these formats, then choose the one most appropriate for your system.

<sup>2</sup> - This format is available starting with V5R1.

### Related concepts

“Control File Transfer Protocol access” on page 16

If you are using File Transfer Protocol (FTP), you need to control users to protect your data and network. This topic offers tips and security considerations.

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

### Related tasks

“Install and register exit programs” on page 14

You can create a library to contain your exit programs and their log files, compile the programs, and register them for use by the File Transfer Protocol (FTP) server.

### Related reference

“Write exit programs for anonymous File Transfer Protocol” on page 12

To use anonymous File Transfer Protocol (FTP), you need to write two exit programs: FTP Server Logon exit program and FTP Server Request Validation exit program.

## Request validation exit point: Client and server

The Request Validation exit points can be used to restrict operations which can be performed by FTP users. Request validation exit points are provided by both the FTP client and server; to restrict both FTP client and FTP server access, exit programs must be added to both exit points.

**Tip:** Because both the FTP client and server exit points share the same exit point format, you can write a single program to handle both.

If you implement anonymous FTP, write your FTP Server Request Validation exit program to restrict anonymous FTP users to retrieve subcommands only, and never allow anonymous users to execute CL commands.

### **What your program should include**

- Exception handling
- Debugging
- Logging

### **Allowed and rejected commands**

The FTP request validation exit program gives you control over whether to accept or reject an operation. Decisions made by exit programs are in addition to any validation that is performed by the FTP client or FTP server application. The FTP client or server application calls the exit program registered for that application each time it processes one of these requests:

- Directory or library creation
- Directory or library deletion
- Setting current directory
- Listing file names
- File deletion
- Sending a file
- Receiving a file
- Renaming a file
- Running a CL command

You might want to set value -1 of parameter 8 (Allow operation) in the VRLQ0100 exit point format to always and unconditionally reject a command.

### **Is there an exit program time-out feature?**

There is no time-out for FTP exit programs. If the exit program has an error or exception that it cannot handle, the FTP server will stop the session.

### **Example programs**

Example programs are available to help you set up anonymous FTP on your server. These examples are for illustration purposes. They do not contain all the features to run on a production system. Use these samples as a starting point to build your own programs. By copying portions of the code from the samples, you can add them to programs that you write yourself. Run the sample programs on a system other than your production system.

#### **Related concepts**

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

#### **Related reference**

“Write exit programs for anonymous File Transfer Protocol” on page 12

To use anonymous File Transfer Protocol (FTP), you need to write two exit programs: FTP Server Logon exit program and FTP Server Request Validation exit program.

### **Example: FTP client or server request validation exit program in CL code:**

This is an example of a simple File Transfer Protocol (FTP) Request Validation exit program. It is written in iSeries Command Language (CL).

This code is not complete, but provides a starting point to help you create your own program for the client or server exit point.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

(Pre formatted text in the following example will flow outside the frame.)





```

/* NOTE: For FTP server, operation 0 is ALWAYS under QTCP profile. */
IF      COND(&OPID = 0) THEN(DO)
  IF      COND(&OPINFO = '9.8.7.6') THEN(CHGVAR +
    VAR(%BINARY(&ALLOWOP)) VALUE(&NOALLOW))
  ELSE    CMD(CHGVAR VAR(%BINARY(&ALLOWOP)) +
    VALUE(&ALLOW))
  GOTO    CMDLBL(END)
ENDDO

/* Check for ANONYMOUS user */
IF      COND(&USRPRF = 'ANONYMOUS ') THEN(DO)
/* Don't allow the following operations for ANONYMOUS user: */
/* 1 (Directory/library creation); 2 (Directory/library deletion); */
/* 5 (File deletion); 7 (Receive file); 8 (Rename file); 9 (Execute CL cmd) */
IF      COND(&OPID = 1 | &OPID = 2 | +
  &OPID = 5 | &OPID = 7 | &OPID = 8 | +
  &OPID = 9) THEN(CHGVAR +
  VAR(%BINARY(&ALLOWOP)) VALUE(&NOALLOW))
ELSE    CMD(DO)
/* For operations 3 (change directory), 4 (list directory) and 6 (send file), */
/* only allow if in PUBLIC library OR "/public" directory. Note that all */
/* path names use the Integrated File System naming format. */
IF      COND(&OPID = 3 | &OPID = 4 | &OPID = 6) THEN(DO)
/* First, convert path name to uppercase (since names in "root" and library */
/* file systems are not case sensitive). */
CALL PGM(QLGCNVCS) PARM(&CASEREQ &OPINFO &PATHNAME +
  &OPLNIN &ERROR)
/* Note: must check for "/public" directory by itself and path names starting */
/* with "/public/". */
IF      COND((%SUBSTRING(&PATHNAME 1 20) *NE +
  '/QSYS.LIB/PUBLIC.LIB') *AND +
  (&PATHNAME *NE '/PUBLIC') *AND +
  (%SUBSTRING(&PATHNAME 1 8) *NE '/PUBLIC/')) +
  THEN(CHGVAR +
  VAR(%BINARY(&ALLOWOP)) VALUE(&NOALLOW))
ELSE    CMD(CHGVAR VAR(%BINARY(&ALLOWOP)) +
  VALUE(&ALLOW))
ENDDO
ENDDO
ENDDO
/* Not ANONYMOUS user: allow everything */
ELSE    CMD(CHGVAR VAR(%BINARY(&ALLOWOP)) +
  VALUE(&ALLOW))

END:    ENDPGM

```

### Example: FTP server request validation exit program in ILE RPG code:

This example demonstrates a simple File Transfer Protocol (FTP) Request Validation Exit program used between the client and the server.

This is an example of a simple FTP Server Request Validation exit program. It is written in ILE RPG programming language. This code is not complete, but provides a starting point to help you create your own program.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

(Pre formatted text in the following example will flow outside the frame.)

```

* Module Description *****
*
*                               PROGRAM FUNCTION
*
* This program demonstrates some of the abilities an FTP Client
* and Server Request Validation Exit Program can have.
*
* Note: This program is a sample only and has NOT undergone any
*       formal review or testing.
*
*****
F/SPACE 3
*****
*
*                               INDICATOR USAGE
*
* IND. DESCRIPTION
*
* LR - CLOSE FILES ON EXIT
*
*****
F/EJECT
*****
* DATA STRUCTURES USED BY THIS PROGRAM
*
* Define constants
*
D Anonym      C          CONST('ANONYMOUS ')
D PublicLib   C          CONST('/QSYS.LIB/ITSOIC400.LIB')
D PublicDir   C          CONST('//ITSOIC.400')
*
* Some CL commands to used later on in the program
*
D ClearSavf   C          CONST('CLRSAVF ITSOIC400/TURVIS')
D SaveLib     C          CONST('SAVLIB LIB(ITSOIC400) -
D              DEV(*SAVF) -
D              SAVF(ITSOIC400/TURVIS)')
*
* A value to be used to trigger a benevolent 'Trojan Horse'
*
D Savetti     C          CONST('ITSOIC400.LIB/TURVIS.FILE')  Extension is FILE
*                                                         although it is a
*                                                         SAVF (and entered as
*                                                         SAVF by the user)
*
* Some nice fields to help us through from lower to upper case character conversion
* 1
D LW          C          CONST('abcdefghijklmnopqrstuvwxy')
D UP          C          CONST('ABCDEFGHIJKLMNOPQRSTUVWXYZ')
*
D NeverAllow  C          CONST(-1)
D DontAllow  C          CONST(0)
D Allow       C          CONST(1)
D AlwaysAllw C          CONST(2)
C/EJECT
*****
* VARIABLE DEFINITIONS AND LISTS USED BY THIS PROGRAM
*****
C/SPACE 2
*
* Define binary parameters
*
D
D APPIDds    DS          1      4B 0
D OPIDds     5          8B 0
D IPLENds    9         12B 0
D OPLENds   13         16B 0
D ALLOWOPds 17         20B 0
*
C *LIKE      DEFINE     APPIDds    APPIDIN
C *LIKE      DEFINE     OPIDds     OPIDIN
C *LIKE      DEFINE     IPLENds    IPLENIN
C *LIKE      DEFINE     OPLENds    OPLENIN

```

```

C   *LIKE      DEFINE  ALLOWOPds  ALLOWOP
*
C   *LIKE      DEFINE  OPINFOIN  OPINFO
*
* Define parameter list
*
C   *Entry     PLIST
* Input parameters:
C       PARM                APPIDIN           Application ID
*                               possible values: 0 = FTP Client Program
*                                               1 = FTP Server Program
C       PARM                OPIDIN           Operation ID
*                               possible values: 0 = Initialize Session
*                                               1 = Create Dir/Lib
*                                               2 = Delete Dir/Lib
*                                               3 = Set Current Dir
*                                               4 = List Dir/Lib
*                                               5 = Delete Files
*                                               6 = Send Files
*                                               7 = Receive Files
*                                               8 = Rename Files
*                                               9 = Execute CL cmd
C       PARM                USRPRF           10           User Profile
C       PARM                IPADDRIN        15           Remote IP Address
C       PARM                IPLENIN         Length of IP Address
C       PARM                OPINFOIN        999          Operation-spec. Info
C       PARM                OPLENIN         Length of Oper. Spec
* Return parameter:
C       PARM                ALLOWOP
*                               possible values: -1 = Never Allow
*                                               (And don't bother
*                                               me with this ops
*                                               in this session)
*                                               0 = Reject Operation
*                                               1 = Allow Operation
*                                               2 = Always Allow Oper.
*                                               (And don't bother
*                                               me with this ops
*                                               in this session)
C/EJECT
*****
* The Main Program
*****
*
C       SELECT
C   APPIDIN  WHENEQ  0
C           EXSR    ClientRqs
C   APPIDIN  WHENEQ  1
C           EXSR    ServerRqs
C           ENDSL
*
C           EVAL    *INLR = *ON
C           RETURN
C/EJECT
*****
* S U B R O U T I N E S
*****
* Here we handle all the FTP Client request validation
*****
C   ClientRqs  BEGSR
*
* Check user profile
*
C           SELECT
*
* Check for 'bad' users who are not allowed to do anything ever
*
C   USRPRF    WHENEQ  'JOEBAD  '
*
C           Z-ADD    NeverAllow  ALLOWOP           Ops not allowed
*
* Check for 'normal' users who are not allowed to do some things
*
C   USRPRF    WHENEQ  'JOENORMAL '

```

```

*
C          SELECT
*
C  OPIDIN   WHENEQ   0          New Connection
C          Z-ADD    Allow      ALLOWOP
*
C  OPIDIN   WHENEQ   1          Create Directory/Lib
C  OPIDIN   OREQ    2          Delete Directory/Lib
C  OPIDIN   OREQ    5          Delete Files
C  OPIDIN   OREQ    7          Receive Files from S
C  OPIDIN   OREQ    8          Rename files
C  OPIDIN   OREQ    9          Execute CL Commands
*
C          Z-ADD    NeverAllow  ALLOWOP      Ops never allowed
*
C  OPIDIN   WHENEQ   3          Set Current Dir
C  OPIDIN   OREQ    4          List Directory/Lib
C  OPIDIN   OREQ    6          Send Files to Server
*
* Extract library and directory names for comparison with allowed areas
*
C  OPLENIN   IFGE    11
C  11        SUBST   OPINFOIN:1  Directory  11
C          ELSE
C  OPLENIN   SUBST(P) OPINFOIN:1  Directory
C          ENDIF
C 1 LW:UP    XLATE   Directory  Directory
*
C  OPLENIN   IFGE    23
C  23        SUBST   OPINFOIN:1  Library    23
C          ELSE
C  OPLENIN   SUBST(P) OPINFOIN:1  Library
C          ENDIF
*
C  Directory IFEQ    PublicDir
C  Library   OREQ    PublicLib
C          Z-ADD    Allow      ALLOWOP
C          ELSE
C          Z-ADD    DontAllow  ALLOWOP
C          ENDIF
*
C          OTHER
C          Z-ADD    DontAllow  ALLOWOP
C          ENDSL
*
* Check for 'cool' users who are allowed to do everything
*
C  USRPRF    WHENEQ   'JOEGOOD  '
C  USRPRF    OREQ    'A960101B '
C  USRPRF    OREQ    'A960101C '
C  USRPRF    OREQ    'A960101D '
C  USRPRF    OREQ    'A960101E '
C  USRPRF    OREQ    'A960101F '
C  USRPRF    OREQ    'A960101Z '
* Allow All FTP Operations
C          Z-ADD    AlwaysAllw  ALLOWOP
*
2 * Any Other User: We leave the back door open and allow
* all operations. If you want to use this program for securing
* your system, then close this door!
*
C          OTHER
C          Z-ADD    AlwaysAllw  ALLOWOP
C*****          Z-ADD    NeverAllow  ALLOWOP
C          ENDSL
*
C          ENDSR
C/EJECT
*****
* Here we handle all the FTP Server request validation *
*****
C  ServerRqs  BEGSR
*
* Check for ANONYMOUS user
*

```

```

C   USRPRF      IFEQ      Anonym
*
C           SELECT
*
C   OPIDIN      WHENEQ     1           Create Directory/Lib
C   OPIDIN      OREQ       2           Delete Directory/Lib
C   OPIDIN      OREQ       5           Delete Files
C   OPIDIN      OREQ       7           Receive Files from C
C   OPIDIN      OREQ       8           Rename files
C   OPIDIN      OREQ       9           Execute CL Commands
*
C           Z-ADD      NeverAllow  ALLOWOP           Ops never allowed
*
C   OPIDIN      WHENEQ     3           Set Current Dir
C   OPIDIN      OREQ       4           List Directory/Lib
C   OPIDIN      OREQ       6           Send Files to Client
*
* Extract library and directory names for comparison with allowed areas
*
C   OPLENIN     IFGE       11
C   11          SUBST      OPINFOIN:1  Directory      11
C           ELSE
C   OPLENIN     SUBST(P)   OPINFOIN:1  Directory
C           ENDIF
C 1 LW:UP       XLATE      Directory  Directory
*
C   OPLENIN     IFGE       23
C   23          SUBST      OPINFOIN:1  Library        23
C           ELSE
C   OPLENIN     SUBST(P)   OPINFOIN:1  Library
C           ENDIF
*
C   Directory   IFEQ       PublicDir           Allowed Directory
C   Library     OREQ       PublicLib           or Library
C           Z-ADD      Allow      ALLOWOP
C           ELSE
C           Z-ADD      DontAllow  ALLOWOP
C           ENDIF
*
C           OTHER
C           Z-ADD      DontAllow  ALLOWOP
C           ENDSL
*
C           ELSE
*
* Any Other User: Allow All FTP Operations
*
C   OPIDIN      IFEQ       6           Send Files to Client
*
* If client issued GET for save file HESSU in library HESSU then we refresh the contents
*
*
C   LW:UP       XLATE      OPINFOIN     OPINFO
C           Z-ADD      0      i           3 0
C   Savetti     SCAN       OPINFO:1     i
*
C   i           IFGT       0
*
* We assume that the save file exists and here clear the save file
*
C           MOVE(L(p)  ClearSavf  Cmd      80
C           Z-ADD      19      Len      15 5
C           CALL      'QCMDEXC'           9999
C           PARM
C           PARM           Cmd
C           PARM           Len
*
* and here we save the library to the save file
*
C           MOVE(L(p)  SaveLib     Cmd
C           Z-ADD      46      Len
C           CALL      'QCMDEXC'           9999
C           PARM
C           PARM           Cmd
C           PARM           Len
C           ENDIF
C           ENDIF

```

```

*
C      Z-ADD      Allow      ALLOWOP
C      ENDIF
*
C      ENDSR

```

**VLRQ0100 exit point format:**

The exit point for FTP Server Application Request Validation is:

QIBM\_QTMF\_SERVER\_REQ

The exit point for FTP Client Application Request Validation is:

QIBM\_QTMF\_CLIENT\_REQ

The interface that controls the parameter format for the exit point is:

VLRQ0100

The following table shows the parameters and parameter format for the VLRQ0100 interface.

**Required parameter format for the VLRQ0100 exit point interface**

Parameter	Description	Input or output	Type and length
1	Application identifier	Input	Binary (4)
2	Operation identifier	Input	Binary (4)
3	User profile	Input	Char (10)
4	Remote IP address	Input	Char (*)
5	Length of remote IP address	Input	Binary (4)
6	Operation-specific information	Input	Char (*)
7	Length of operation-specific information	Input	Binary (4)
8	Allow operation	Output	Binary (4)

Here are the parameter descriptions:

**VLRQ0100 Parameter 1:**

Application identifier

**INPUT; BINARY(4)**

Identifies the TCP/IP application program that is making the request. Four different TCP/IP applications share the VLRQ0100 interface. The first parameter identifies which application is calling the exit program. The following table demonstrates the possible values.

Value	Application
0	FTP client program
1	FTP server program
2	REXEC server program
3	TFTP server program

**VLRQ0100 Parameter 2:**

Operation identifier

**Input; Binary(4)**

Indicates the operation (command) that the FTP user wants (requests) to perform.

The following table demonstrates the possible values when the application identifier (parameter 1) indicates the FTP client or FTP server program.

Value	Operation ID	Client subcommand	Server subcommand
0	Start session	Open, SECOpen	New connection
1	Create directory/library	*	MKD, XMDK
2	Delete directory/library	*	RMD, XRMD
3	Set current directory/library	LCD	CWD, CDUP, XCWD, XCUP
4	List files	*	LIST, NLIST
5	Delete file	*	DELE
6	Send file	APPEND, PUT, MPUT	RETR
7	Receive file	GET, MGET	APPE, STOR, STOU
8	Rename file	*	RNFR, RNT0
9	Execute CL command	SYSCMD	RCMD, ADDm, ADDV, CRTL, CRTP, CRTS, DLTF, DTLT

**Note:** The asterisk symbol (\*) represents control operations that the FTP client exit does not recognize. The only way a client can use these operations is with CL commands using the FTP client subcommand SYSCMD. Operation identifier 9 controls the execution of CL commands.

**VLRQ0100 Parameter 3:**

User profile

**INPUT; Char(10)**

The user profile for the FTP session.

**VLRQ0100 Parameter 4:**

Remote IP address

**INPUT; CHAR(\*)**

The Internet Protocol (IP) address of the remote host system. The format for this string is dotted decimal (123.45.67.89), left-aligned. The remote host can be a client or a server that is based on the setting of the application identifier parameter.

**VLRQ0100 Parameter 5:**

Length (in bytes) of the remote IP address (parameter 4)

**INPUT; BINARY(4)**

The length of the remote IP address (parameter 4).

**VLRQ0100 Parameter 6:**

Operation-specific information

**INPUT; CHAR(\*)**

Information that describes the requested operation. The contents of this field depend on the values of the operation identifier (parameter 2), and the application identifier (parameter 1). For example:



**For operation identifier 0 and application identifier 0**

There is no operation-specific information. This field is blank.

**For operation identifier 0 and application identifier 1**

The operation-specific information contains the IP address of the TCP/IP interface that connects to the local host (FTP server) for this session. The format for this string is dotted decimal (123.45.67.89), left-aligned.

**For operation identifiers 1 through 3**

The operation-specific information contains the name of the directory or library in which to perform the operation. The format for the directory or library name is an absolute path name.

**For operation identifiers 4 through 8**

The operation-specific information contains the name of the file on which to perform the operation. The format for the file name is an absolute path name.

**For operation identifier 9**

The operation-specific information contains the iSeries Control Language (CL) command the user requests.

**VLRQ0100 Parameter 7:**

Length of operation-specific information.

**INPUT; BINARY(4)**

Indicates the length of the operation-specific information (parameter 6). Length is 0 when the exit point does not provide operation-specific information.

**VLRQ0100 Parameter 8:**

Allow operation.

**OUTPUT; BINARY(4)**

Indicates whether to allow or reject the requested operation. The following table demonstrates the possible values.

Value	Description
-1	<i>Never</i> allow this operation identifier:  Reject this operation identifier unconditionally for the remainder of the current session.  This operation identifier will not call the exit program again.
0	Reject the operation
1	Allow the operation
2	<i>Always</i> allow this operation identifier:  Allow this operation identifier unconditionally for the remainder of the current session.  This operation identifier will not call the exit program again.

**Related concepts**

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

*VLRQ0100 exit point format usage notes:*

VLRQ0100 is the exit point format that is used for both the File Transfer Protocol (FTP) Client Request Validation Exit Point and the FTP Server Request Validation Exit Point.

## Incorrect output parameters

If the output returned for the Allow Operation parameter (parameter 8) is not valid, then the FTP server rejects the requested operation and posts this message to the job log:

*Data from exit program for exit point &1 is missing or not valid*

## Exceptions

If the FTP server encounters any exception when calling the exit program, it posts this message to the job log:

*Exception encountered for FTP exit program &1 in library &2 for exit point &3*

## Summary: Operation-specific information

This table summarizes the Operation-specific information (VLRQ0100 parameter 6) that is required for each Operation identifier (VLRQ0100 parameter 2).

Operation identifier (VLRQ0100 Parm 2)	Operation-specific information (VLRQ0100 parameter 6)
0	NONE if application ID=0 (parameter 1)
0	Dotted decimal format IP address of client host when application ID=1 or 2 (parameter 1)
1-3	Absolute path name of library or directory. Examples:  /QSYS.LIB/QGPL.LIB <sup>(a)</sup> /QOpenSys/DirA/DirAB/DirABC <sup>(b)</sup>
4-8	Absolute path name of file. Examples:  /QSYS.LIB/MYLIB.LIB/MYFILE.FILE/MYMEMB.MBR <sup>(a)</sup> /QOpenSys/DirA/DirAB/DirABC/FileA1 <sup>(b)</sup>

### Notes:

<sup>(a)</sup> - QSYS.LIB file system path names are always in uppercase

<sup>(b)</sup> - QOpenSys file system path names are case sensitive and might include upper and lower case letters.

## Server logon exit point

You can control the authentication of users to a TCP/IP application server with the TCP/IP Application Server Logon exit point.

This exit point allows server access based on the originating session's address. It also allows you to specify an initial working directory that is different from those that are in the user profile.

When you add an exit program to the exit point, the server calls the logon exit program each time a user attempts to log on. The exit program sets the return code output parameter to indicate whether the server will continue the logon operation. Alternate return code settings are available for processing the logon, and initializing directory information.

The iSeries exit point for FTP server logon is:

QIBM\_QTMF\_SVR\_LOGON

These are the three exit point formats available:

- The TCPL0100 exit point format allows this basic logon control:
  - Ability to accept or reject a logon
  - Control of the user profile, password, and current library
- The TCPL0200 exit point format provides additional parameters to control the logon process, including:
  - Ability to set the working directory to any directory on the system.
  - Ability to return application-specific information
  - Ability to control encryption of FTP data sent to and received from the FTP client.
- The TCPL0300 exit point format extends the TCPL0200 format, so you can use i5/OS enhanced password support and the additional parameters to enable CCSID processing for password and directory name fields. In addition, when the user for the session has been authenticated with a client certificate, the exit program receives the client certificate.

**Notes:**

1. There can be only one exit program registered for the FTP server logon exit point. You must decide which of the three exit point formats you want to use.
2. For the FTP application, this exit point provides the capability to implement anonymous FTP, including the information required to log and control access.
3. For all character parameters in exit point formats TCPL0100 and TCPL0200, and all character parameters without an associated CCSID in exit point format TCPL0200: Character data passed to the exit program is in the CCSID of the job. If the job CCSID is 65535, the character data is in the default CCSID of the job. Any character data that is returned by the exit program in these parameters is expected to be in this same CCSID.

## **Server logon exit program for anonymous FTP**

For anonymous FTP, write the server logon program to perform the following functions:

- Accept logons from user ID ANONYMOUS
- Request an e-mail address as a password. It is customary to require a 'valid e-mail address' for the password. The term is misleading because the exit program only verifies if there is an '@' symbol in the middle of a string of alphanumeric characters. That too is customary. This is why it is important to log the user's IP address.
- Check for the @ symbol in the password string.
- Force ANONYMOUS users to your public access library only. See return code 3 of parameter 8 for TCPL0200 Format).

### **What your program should include:**

- Exception handling
- Debugging
- Logging
  - Log the IP address and e-mail address (sent as a password) of the FTP requester.

### **Is there an exit program time-out feature?**

There is no time-out for FTP exit programs. If the exit program has an error or exception that it cannot handle, the FTP server will abort the session.

## QTCP needs authority

When the application calls the FTP server logon exit program, the FTP server job is running under the QTCP user profile.

Make sure that QTCP has sufficient authority to access and write to any log files or other satellite files associated with the exit programs.

## Example programs

Example programs are available to help you set up anonymous FTP on your server. These examples are for illustration purposes. They do not contain enough features to run on a production machine as is. You can use these samples as a starting point to build your own programs. By copying portions of the code from the samples, you can add them to programs that you write yourself. It is recommended that you run the sample programs on a system other than your production system.

### Related concepts

“Control File Transfer Protocol access” on page 16

If you are using File Transfer Protocol (FTP), you need to control users to protect your data and network. This topic offers tips and security considerations.

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

“Configure anonymous File Transfer Protocol” on page 10

Anonymous File Transfer Protocol (FTP) enables remote users to use the FTP server without an assigned user ID and password.

“Determine problems with File Transfer Protocol” on page 143

You can view a list of steps to determine whether your Simple Mail Transfer Protocol (SMTP) is working correctly.

### Related reference

“Write exit programs for anonymous File Transfer Protocol” on page 12

To use anonymous File Transfer Protocol (FTP), you need to write two exit programs: FTP Server Logon exit program and FTP Server Request Validation exit program.

### Example: FTP server logon exit program in CL code:

This is an example of a simple File Transfer Protocol (FTP) Server Logon exit program. It is written in iSeries Command Language (CL).

This is an example of a simple FTP Server Logon exit program. It is written in iSeries Command Language (CL). This code is not complete, but provides a starting point to help you create your own program.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

(Pre formatted text in the following example will flow outside the frame.)

```

/*****/
/*
/* Sample FTP server logon exit program.
/* Note: This program is a sample only and has not undergone any formal
/* review or testing.
/*
/* Additional notes:
/* 1. When the FTP server logon exit is called, the FTP server job is
/* running under the QTCP user profile.
/* 2. For the ANONYMOUS case, users can add logging capability (for
/* example, write the E-mail address entered for the password and
/* the client IP address to a log file).
/* 3. IBM strongly recommends that you create the exit program in a library
/* with *PUBLIC authority set to *EXCLUDE, and give the exit program
/* itself a *PUBLIC authority of *EXCLUDE. The FTP server adopts
/* authority when it is necessary to resolve and call the exit program.
/*
/*****/

TSTLOGCL: PGM          PARM(&APPIDIN &USRIN &USRLENIN &AUTIN &AUTLENIN +
          &IPADDRIN &IPLLENIN &RETCDOUT &USRPRFOUT &PASSWDOUT +
          &CURLIBOUT)

/* Declare input parameters */
DCL      VAR(&APPIDIN)  TYPE(*CHAR) LEN(4) /* Application identifier */
DCL      VAR(&USRIN)    TYPE(*CHAR) LEN(999)/* User ID */
DCL      VAR(&USRLENIN) TYPE(*CHAR) LEN(4) /* Length of user ID */
DCL      VAR(&AUTIN)    TYPE(*CHAR) LEN(999)/* Authentication string */
DCL      VAR(&AUTLENIN) TYPE(*CHAR) LEN(4) /* Length of auth. string */
DCL      VAR(&IPADDRIN) TYPE(*CHAR) LEN(15) /* Client IP address */
DCL      VAR(&IPLLENIN) TYPE(*CHAR) LEN(4) /* IP address length */
DCL      VAR(&RETCDOUT) TYPE(*CHAR) LEN(4) /* return code (out) */
DCL      VAR(&USRPRFOUT) TYPE(*CHAR) LEN(10) /* user profile (out) */
DCL      VAR(&PASSWDOUT) TYPE(*CHAR) LEN(10) /* password (out) */
DCL      VAR(&CURLIBOUT) TYPE(*CHAR) LEN(10) /* current library (out) */

/* Declare local copies of parameters (in format usable by CL) */
DCL      VAR(&APPID)    TYPE(*DEC) LEN(1 0)
DCL      VAR(&USRLEN)   TYPE(*DEC) LEN(5 0)
DCL      VAR(&AUTLEN)   TYPE(*DEC) LEN(5 0)
DCL      VAR(&IPLLEN)   TYPE(*DEC) LEN(5 0)

/* Assign input parameters to local copies */
CHGVAR   VAR(&APPID)    VALUE(%BINARY(&APPIDIN))
CHGVAR   VAR(&USRLEN)   VALUE(%BINARY(&USRLENIN))
CHGVAR   VAR(&AUTLEN)   VALUE(%BINARY(&AUTLENIN))
CHGVAR   VAR(&IPLLEN)   VALUE(%BINARY(&IPLLENIN))

/* Check for ANONYMOUS user. Allow for ANONYMOUSA, etc. as "regular" */
/* user profile. */
IF       COND(&USRLEN = 9) THEN(DO)
IF       COND(%SST(&USRIN 1 9) = 'ANONYMOUS')
THEN(DO)
/* For anonymous user: want to force user profile ANONYMOUS current library to PUBLIC. */
CHGVAR  VAR(%BINARY(&RETCDOUT)) VALUE(6)
CHGVAR  VAR(&USRPRFOUT) VALUE('ANONYMOUS ')
CHGVAR  VAR(&CURLIBOUT) VALUE('PUBLIC ')
ENDDO
/* Any other user: proceed with normal logon processing. */
ELSE    CMD(CHGVAR VAR(%BINARY(&RETCDOUT)) VALUE(1))
ENDDO
ELSE    CMD(CHGVAR VAR(%BINARY(&RETCDOUT)) VALUE(1))

END:      ENDPGM

```

### Example: FTP server logon exit program in C code:

This is an example of a simple File Transfer Protocol (FTP) Server Logon exit program. It is written in C programming language.

This code is not complete, but provides a starting point to help you create your own program.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

(Pre formatted text in the following example will flow outside the frame.)

```

/* Module Description *****/
/*
/*****/
/*
/* Note: This program is a sample only and has NOT undergone any
/ formal review or testing.
/*
/*
/*****/
/*
/* Source File Name: qtmfsvrlgn.c
/*
/*
/* Module Name: FTP Server Logon exit program.
/*
/*
/* Service Program Name: n/a
/*
/*
/* Source File Description:
/* This example exit program provides additional control over the
/* process of authenticating a user to a TCP/IP application server.
/* When installed, this example exit program would be called each
/* time a user attempts to log on to the server.
/*
/*****/
/*
/* Function List: main - FTP Server Logon exit program main.
/* qtmfsvrlgn - FTP Server Logon exit function.
/* CheckClientAddress - Check originating sessions IP
/* address.
/*
/*
/* End Module Description *****/
#define _QTMFSVRLGN_C

/*****/
/* All file scoped includes go here
/*****/
#ifndef __stdio_h
#include <stdio.h>
#endif

#ifndef __ctype_h
#include <ctype.h>
#endif

#ifndef __string_h
#include <string.h>
#endif

#ifndef __stdlib_h
#include <stdlib.h>
#endif

#include "qusec.h" /* Include for API error code structure */
#include "qsyrusri.h" /* Include for User Information API */

/*****/
/* All file scoped Constants go here
/*****/
#define EQ ==
#define NEQ !=
#define BLANK ' '
#define FWIDTH 128 /* Width of one database file record */
#define FNAME 21 /* Qualified database file name width */

/* Valid characters for Client IP address. The CheckClientAddress()
/* function will check the Client IP address input argument
/* (ClientIPaddr_p) to ensure it is in valid dotted-decimal format.
/* This is one example of an input validity check.
const char ValidChars[] = "0123456789.";
/*****/
/* All file scoped type declarations go here
/*****/

```

```

/*****/
/* All file-scoped macro calls go here */
/*****/

/*****/
/* All internal function prototypes go here */
/*****/

static void qtmfsvrlgn
    (int,char *,int,char *,int,char *,int,int *,char *,char *,char *);

static int CheckClientAddress(char *, int);

/*****/
/* All file scoped variable declarations go here */
/*****/

/*****/
/*          ** NOTE **          */
/* The following client IP address are for example purposes only. Any */
/* resemblance to actual system IP addresses is purely coincidental. */
/*****/

/* EXCLUSIVE system lists, ie - Logon attempts from any client IP */
/* addresses NOT in one of these lists */
/* are allowed to continue. */
/* Reject server logon attempts of users attempting to log in from */
/* these client systems (return code = 0) */
char Reject[] = "1.2.3.4 5.6.7.8";
/* Limit logon abilities of users attempting to log in as ANONYMOUS */
/* from these client systems (return code = 6). */
/* In this example program, the initial current library is set and */
/* returned as an output parameter for users attempting to log in */
/* as ANONYMOUS from these specific client systems. */
char Limit[] = "9.8.7.6 4.3.2.1 8.7.6.5";

/* Function Specification *****/
/* */
/* Function Name: Main */
/* */
/* Descriptive Name: FTP Server Logon exit program main. */
/* */
/* This example exit program allows access to a TCP/IP server to */
/* be controlled by the address of the originating session, gives */
/* additional control over the initial current library to a user, */
/* and provides the capability to implement "anonymous" FTP. */
/* */
/* Notes: */
/* */
/* Dependencies: */
/* FTP Server Logon exit point QIBM_QTMF_SVR_LOGON was registered */
/* during FTP product installation. */
/* */
/* Restrictions: */
/* */
/* None */
/* */
/* Messages: */
/* */
/* None */
/* */
/* Side Effects: */
/* */
/* None */
/* */
/* Functions/Macros called: */
/* */
/* qtmfsvrlgn - Server Logon exit function. */
/* */

/* Input: */

```



```

/* int * argv[1] - Identifies requesting application */
/* (FTP Client =0, FTP Server = 1). */
/* char * argv[2] - User identifier from client program. */
/* (For FTP server, this is user CMD data */
/* int * argv[3] - Length (in bytes) of User ID string. */
/* char * argv[4] - Authentication string from client. */
/* (For FTP server, this is the password) */
/* int * argv[5] - Length (bytes) Authentication string. */
/* char * argv[6] - Internet Protocol address from which */
/* the session originates. */
/* int * argv[7] - Length (in bytes) of IP address. */
/* int * argv[8] - Return code (received as 0). */
/* char * argv[9] - User profile (received as blanks). */
/* char * argv[10] - Password (received as blanks). */
/* char * argv[11] - Initial current library (received as blanks)*/
/*
/* Exit Normal: Return Return Code, User Profile, Password, Initial */
/* Current Library to server application. */
/*
/* Exit Error: None */
/*
/* End Function Specification *****/
void main(int argc, char *argv[])
{
/*****/
/* Code */
/*****/

/*****/
/* Collect input arguments and call function to determine if client */
/* should be allowed to log in to an FTP server application. */
/*****/
qtmfsvrign(*(int *)argv[1]), /* Application Identifier
(Input) */
        argv[2], /* User Identifier (Input) */
        *((int *)argv[3]), /* Length User of
Identifier(Input) */
        argv[4], /* Authentication String (Input) */
        *((int *)argv[5]), /* Length of Authentication string */
(Input) */
        argv[6], /* Client IP Address (Input) */
        *((int *)argv[7]), /* Length of Client IP Address */
(Input) */
        (int *)argv[8]), /* Return Code (Output)*/
        argv[9], /* User Profile (Output)*/
        argv[10], /* Password (Output)*/
        argv[11]); /* Initial Current Library (Output)*/
return;
}

/* Function Specification *****/
/*
/* Function Name: qtmfsvrign */
/*
/* Descriptive Name: Server Logon exit function. */
/*
/* This exit function provides control over user authentication to */
/* an FTP server. */
/*
/* Notes: */
/*
/* Dependencies: */
/*
/* FTP Server Logon exit point QIBM_QTMF_SVR_LOGON was */
/* registered during FTP product installation. */
/*
/* Restrictions: */
/*
/* None */
/*
/* Messages: */
/*
/* None */

```

```

/*                                                                    */
/* Side Effects:                                                         */
/*                                                                    */
/*     None                                                             */
/*                                                                    */
/* Functions/Macros called:                                             */
/*                                                                    */
/*     CheckClientAddress - Check the ClientIPAddr_p input argument.*/
/*     memcpy  - Copy bytes from source to destination.                */
/*     memset  - Set bytes to value.                                    */
/*     strstr  - Locate first occurrence of substring.                  */
/*     sprintf - Formatted print to buffer.                             */
/*                                                                    */
/* Input:                                                                */
/*     int  ApplId      - Application Identifier (Server = 1).          */
/*     char * UserId_p  - User identifier from client program.         */
/*                       (For FTP server, USER subcommand data)*/
/*     int  Lgth_UserId - Length (in bytes) of user ID string.         */
/*     char * AuthStr_p - Authentication string from client.           */
/*                       (For FTP server, this is the password)*/
/*     int  Lgth_AuthStr - Length (bytes) Authentication string.       */
/*     char * ClientIPAddr_p - Internet Protocol address from which   */
/*                               the session originates.                */
/*     int * Lgth_ClientIPAddr - Length (in bytes) of IP address.     */
/*                                                                    */

/* Output:                                                                */
/*     int * ReturnCode: Indicates degree of success of operation:     */
/*     ReturnCode = 0 - Reject logon.                                    */
/*     ReturnCode = 1 - Continue logon; use initial current library*/
/*     ReturnCode = 2 - Continue logon; override initial current      */
/*                       library                                       */
/*     ReturnCode = 3 - Continue logon; override user, password       */
/*     ReturnCode = 4 - Continue logon; override user, password,     */
/*                       current library                               */
/*     ReturnCode = 5 - Accept logon; override user profile           */
/*     ReturnCode = 6 - Accept logon; override user profile,         */
/*                       current library                               */
/*     char * UserProfile - User profile to use for this session       */
/*     char * Password    - Password to use for this session          */
/*     char * Init_Cur_Lib - Initial current library for this session  */
/*                                                                    */
/* Exit Normal: (See OUTPUT)                                           */
/*                                                                    */
/* Exit Error: None                                                    */
/*                                                                    */
/* End Function Specification *****/
static void qtmfsvrln(int ApplId, /* Entry point */
                    char *UserId_p,
                    int Lgth_UserId,
                    char *AuthStr_p,
                    int Lgth_AuthStr,
                    char *ClientIPAddr_p,
                    int Lgth_ClientIPAddr,
                    int *ReturnCode,
                    char *UserProfile_p,
                    char *Password_p,
                    char *InitCurrLib_p)
{
    /******
    /* Local Variables                                                    */
    /******
    /* The following lists serve as an example of an additional layer   */
    /* of control over user authentication to an application server.     */
    /* Here, logon operations using the following user identifiers       */
    /* will be allowed to continue, but the output parameters returned  */
    /* by this example exit program will vary depending on which list   */
    /* a user identifier (UserId_p) is found in.                          */
    /* For example, attempts to logon as FTPUSR1 or FTPUSR2 will be     */
    /* allowed, and this example exit will return the initial current   */
    /* library as an output parameter along with a return code of 2.    */
    /******
    /* Continue the logon operation, Return Code = 1                    */
    char Return1[] = "FTPUSR10 ";

```

```

/* Continue the logon operation, Return Code = 2          */
char Return2[] = "FTPUSR11 FTPUSR2 ";
/* Continue the logon operation, Return Code = 3          */
char Return3[] = "FTPUSR12 FTPUSR3 FTPUSR23 ";
/* Continue the logon operation, Return Code = 4          */
char Return4[] = "FTPUSER FTPUSR4 FTPUSR24 FTPUSR94 ";
int rc; /* Results of server logon request */
Qsy_USRI0300_T Receiver_var; /* QSYRUSRI API Receiver variable */
int Lgth_Receiver_var; /* Receiver variable length */
char Format_Name[8]; /* Format name buffer */
char User_Id[10]; /* User Identifier buffer */
Qus_EC_t error_code = /* QSYRUSRI API error code structure: */
{
    sizeof(Qus_EC_t), /* Set bytes provided */
    0, /* Initialize bytes available */
    ' ',' ',' ',' ',' ',' ',' ',' ', /* Initialize Exception Id */
};
char *pcTest_p; /* Upper-case User Identifier pointer*/
int i; /* "For" loop counter variable */

/*****
/* Code
*****/

/* Test validity of application ID input argument. */
if(1 NEQ ApplId)
{
    /* ERROR - Not FTP server application. */
    /* Return Code of 0 is used here to indicate */
    /* that an incorrect input argument was received. */
    /* The server logon operation will be rejected. */
    rc = 0; /* Application ID not valid */
} /* End If the application identifier is NOT for FTP server */
else /* FTP server application identifier */
{
    /* Validate the client IP address input argument. */
    rc = CheckClientAddress(ClientIPAddr_p,
        Lgth_ClientIPAddr);
    if(0 NEQ rc) /* Valid, acceptable client address */
    {
        /* Initialize User_Id; used to hold upper-cased user identifier */
        memset(User_Id, BLANK, sizeof(User_Id));

        /* Initialize pcTest_p to point to UserId_p input argument. */
        pcTest_p = UserId_p;

        /* Uppercase all of the user ID to compare for ANONYMOUS user. */
        for(i = 0; i < Lgth_UserId; i++)
        {
            User_Id[i] = (char)toupper(*pcTest_p);
            pcTest_p += 1;
        }

        /* If user has logged in as ANONYMOUS. */
        if(0 == memcmp("ANONYMOUS ", User_Id, 10))
        {
            /* Determine how to continue with ANONYMOUS logon attempt. */
            if(NULL NEQ strstr(Limit, ClientIPAddr_p))
            {
                /* If users system IP address is found in the "Limit" list, */
                /* return ReturnCode of 6, user profile and initial */
                /* current library values as output parameters. */
                memcpy(UserProfile_p, "USERA1 ", 10);
                memcpy(InitCurrLib_p, "PUBLIC ", 10);
                rc = 6;
            }
        }
        else
        {
            /* Users system IP address is NOT found in the "Limit" list,*/
            /* return ReturnCode of 5, user profile output parameter; */
            /* use the initial current library that is specified by the */
            /* user profile information. */
            memcpy(UserProfile_p, "USERA1 ", 10);

```

```

        rc = 5;
    }
} /* End If USER is ANONYMOUS */

else /* Else USER is not ANONYMOUS */
{
    /* Set receiver variable length. */
    Lgth_Receiver_var = sizeof(Qsy_USRI0300_T);
    /* Set return information format. */
    memcpy(Format_Name, "USRI0300", sizeof(Format_Name));
    /* Set user identifier passed in. */
    memset(User_Id, BLANK, sizeof(User_Id));
    memcpy(User_Id, UserId_p, Lgth_UserId);
    /* Call QSYRUSRI - Retrieve User Information API */
    QSYRUSRI(&Receiver_var, /* Return Information receiver var */
            Lgth_Receiver_var, /* Receiver variable length */
            Format_Name, /* Return information format name */
            User_Id, /* User ID seeking information */
            &error_code); /* Error return information */
    /* Check if an error occurred (byte_available not equal 0) */
    if(0 NEQ error_code.Bytes_Available)
    {
        /* Return ReturnCode of 0 only (Reject logon); */
        rc = 0; /* Reject the logon operation */
        *ReturnCode = rc; /* Assign result to ReturnCode */
    }
    else /* No error occurred from Retrieve User Info */
    {
        /* (Bytes_Available = 0) */
        /* Set current library for user profile. */
        memcpy(InitCurrLib_p, Receiver_var.Current_Library, 10);
        if(NULL NEQ strstr("CRTDFT",
            Receiver_var.Current_Library))
        {
            memcpy(InitCurrLib_p, "FTPDEFAULT", 10);
        }
    }
    else
    {
        if(NULL NEQ strstr(Return1, UserId_p))
        {
            /* Return ReturnCode of 1 (Continue logon); */
            /* Also return user profile and password output */
            /* parameters to endure they are ignored by the server.*/
            memcpy(UserProfile_p, UserId_p, Lgth_UserId);
            memcpy>Password_p, AuthStr_p, Lgth_AuthStr);
            rc = 1; /* Continue the logon operation */
        }
    }
    else
    {
        if(NULL NEQ strstr(Return2, UserId_p))
        {
            /* Return ReturnCode of 2, and initial current library*/
            /* Also return user profile and password values */
            /* even though they will be ignored by the server. */
            memcpy(UserProfile_p, UserId_p, Lgth_UserId);
            memcpy>Password_p, AuthStr_p, Lgth_AuthStr);
            memcpy(InitCurrLib_p, "FTPEXT2",
                strlen("FTPEXT2"));
            rc = 2; /* Continue logon; return InitCurLib */
        }
    }
    else
    {
        if(NULL NEQ strstr(Return3, UserId_p))
        {
            /* Return ReturnCode of 3, user profile, password. */
            /* Also return initial current library value, */
            /* even though it will be ignored. */
            memcpy(UserProfile_p, UserId_p, Lgth_UserId);
            memcpy>Password_p, AuthStr_p, Lgth_AuthStr);

            memcpy(InitCurrLib_p, "FTPEXT3",
                strlen("FTPEXT3")); /* Server ignores */
        }
    }
}

```



```

/*          0 = Reject the logon operation.          */
/*          ClientIPAddr_p is one that is not      */
/*          allowed, or contains a character       */
/*          that is not valid.                     */
/*          1 = Continue the logon operation.      */
/*          */
/* Exit Normal: (See OUTPUT)                       */
/*          */
/* Exit Error: None.                              */
/*          */
/* End Function Specification *****/

static int CheckClientAddress(char *ClientIPAddr_p, /* Entry point */
                             int Lgth_ClientIPAddr)
{
/*****/
/* Local Variables                               */
/*****/
int rc; /* Return code */

/*****/
/* Code                                          */
/*****/

/* Check that client IP address input argument is dotted-decimal */
/* format of minimum length, with no leading blanks or periods, */
/* and contains only valid characters.                          */
if((Lgth_ClientIPAddr < 7) || /* Minimum IP address size */
   (strspn(ClientIPAddr_p, ValidChars) < Lgth_ClientIPAddr)||
   (strspn(ClientIPAddr_p, ".") EQ 1)|| /* Leading '.' in IP */
   (strspn(ClientIPAddr_p, " ") EQ 1)) /* Leading blank in IP */
   {
/* Client's IP address not valid, or contains an incorrect character */
rc = 0; /* Client IP address input argument not valid */
}
else
{
/* Is client system allowed to log in to FTP server? */
if(NULL NEQ strstr(Reject, ClientIPAddr_p))
{
/* Return code = 0 - Reject the server logon operation, as the */
/* client IP address is found in the global */
/* "Reject" list. */
rc = 0; /* Reject the logon operation */
}
else
{
/* Continue the server logon operation checks. */
rc = 1; /* Continue the logon operation */
}
}
return(rc);
}

#undef _QTMFSVRLGN_C

```

### Example: FTP server logon exit program in ILE RPG code:

This is an example of a simple File Transfer Protocol (FTP) Server Logon exit program. It is written in ILE RPG.

This code is not complete, but provides a starting point to help you create your own program.

**Note:** By using the code examples, you agree to the terms of the “Code license and disclaimer information” on page 151.

(Pre formatted text in the following example will flow outside the frame.)

```

* Module Description *****
*
*****
*
* Note: This program is a sample only and has NOT undergone any
*       formal review or testing.
*
*****
*
*               PROGRAM FUNCTION
*
* This program demonstrates some of the abilities an FTP Server
* Logon Exit Program can have.
*
*****
F/SPACE 3
*****
*
*               INDICATOR USAGE
*
*   IND.  DESCRIPTION
*
*   LR - CLOSE FILES ON EXIT
*
*****
F/EJECT
*****
* DATA STRUCTURES USED BY THIS PROGRAM
*****
*
* Define constants
*
1 D Anonym      C          CONST('ANONYMOUS ')
  D Text1      C          CONST('Anonymous ('
  D Text2      C          CONST(') FTP logon')
  D InvalidNet C          CONST('10.')
C/EJECT
*****
* VARIABLE DEFINITIONS AND LISTS USED BY THIS PROGRAM
*****
C/SPACE 2
*
* Define binary parameters
*
D          DS
D APPIDds      1      4B 0
D USRLEnds    5      8B 0
D AUTLEnds    9     12B 0
D IPLEnds    13     16B 0
D RETCDds    17     20B 0
*
C *LIKE      DEFINE  APPIDds      APPIDIN
C *LIKE      DEFINE  USRLEnds    USRLENIN
C *LIKE      DEFINE  AUTLEnds    AUTLENIN
C *LIKE      DEFINE  IPLEnds    IPLENIN
C *LIKE      DEFINE  RETCDds    RETCDOUT
*
* Define parameter list
*
C *Entry      PLIST
* Input parameters:
C          PARM          APPIDIN          Application ID
*                                     possible values: 1 = FTP Server Program
C          PARM          USRIN            999          User ID
C          PARM          USRLENIN        999          Length of User ID
C          PARM          AUTIN            999          Authentication Strg
C          PARM          AUTLENIN        999          Length of Auth. Strg
C          PARM          IPADDRIN        15           Client IP Address
C          PARM          IPLENIN        15           Length of IP Address
* Return parameters:
C          PARM          RETCDOUT        0            Return Code (Out)
*                                     possible values: 0 = Reject Logon
*                                               1 = Continue Logon
*                                               2 = Continue Logon,

```





## TCPL0100 exit point format:

Exit Point Format Name: TCPL0100

Exit Point Name: QIBM\_QTMF\_SVR\_LOGON

Exit Point Name: QIBM\_QTMX\_SVR\_LOGON

This is the required parameter group.

Parameter	Description	Input or output	Type and length
1	Application identifier	Input	Binary(4)
2	User identifier	Input	Char(*)
3	Length of user identifier	Input	Binary(4)
4	Authentication string	Input	Char(*)
5	Length of authentication string	Input	Binary(4)
6	Client IP address	Input	Char(*)
7	Length of client IP address	Input	Binary(4)
8	Return code	Output	Binary(4)
9	User profile	Output	Char(10)
10	Password	Output	Char(10)
11	Initial current library	Output	Char(10)

## Parameter descriptions

### Application identifier

INPUT; BINARY(4) Identifies the requested application server. The valid values are:

- 1 FTP server program
- 2 REXEC server program

### User identifier

INPUT; CHAR(\*) The user identification supplied by the client program. For the FTP server, this parameter contains the data field from the USER subcommand.

### Length of user identifier

INPUT; BINARY(4) The length (in bytes) of the user identifier string.

### Authentication string

INPUT; CHAR(\*) The string (such as a password) supplied by the client program.

For the FTP server, this parameter contains the data field from the PASS (password) subcommand. Beginning with V5R1, if the user is authenticated by a client certificate, no data is provided for this parameter.

### Length of authentication string

INPUT; BINARY(4) The length (in bytes) of the authentication string.

**Note:** For the FTP server: When the user is authenticated by a client certificate, this parameter is set to 0.

### Client IP address

INPUT; CHAR(\*) The Internet Protocol (IP) address from which the session originates. This string is in dotted decimal format, left justified.

### Length of client IP address

INPUT; BINARY(4) Indicates the length (in bytes) of the client IP address.

### Return code

OUTPUT; BINARY(4) Indicates whether to accept or reject the logon operation, to perform password authentication, and whether or not to override the initial current library. The valid values are:

- 0 Reject the logon operation. Ignore the user profile, password, and initial current library output parameters.
- 1 Continue the logon operation with the specified user identifier and authentication string, and the user-specified the initial current library. The user identifier becomes the user profile, and the authentication string becomes the password. The program ignores the user profile, password, and initial current library output parameters.

**Note:** For the logon to succeed, the authentication string must match the user profile-specified password.

- 2 Continue the logon operation with the specified user identifier and authentication string, and override the initial current library with the one specified by the initial current library parameter. The user identifier is the user profile. The authentication string is the password. Provide the initial current library output parameter. The program ignores the user profile and password output parameters.

**Note:** For the logon to succeed, the authentication string must match the user profile-specified password.

- 3 Continue the logon operation. Override the user profile and password with those values you received from the output parameters of this exit program. Use the user profile-specified initial current library that the exit program returns. The program ignores the initial current library output parameter.

**Note:** For the logon to succeed, the password output parameter must match the user profile-specified password.

*Attention!* IBM strongly recommends that you **never** code passwords directly in an exit program. Encryption, for example, allows algorithmic password determination.

- 4 Continue the logon operation, which will override the user profile, password, and initial current library with output parameters of this exit program.

**Note:** For the logon to succeed, the password output parameter must match the user profile-specified password.

*Attention!* IBM strongly recommends that you **never** code passwords directly in an exit program. Encryption, for example, allows algorithmic password determination.

- 5 Accept the logon operation. Override the user profile is returned in the user profile output parameter of this exit program. Use the initial current library specified by the user profile, returned by this exit program. The program ignores the output parameters for the initial current library and password.

**Note:** Specifying this value overrides normal i5/OS password processing. It is the only password authentication.

- 6 Accept the logon operation. Override the user profile and initial current library with those that are returned in the output parameters of this exit program. Ignore the output parameter for password.

**Note:** Specifying this value overrides normal i5/OS password processing. It is the only password authentication.

**User profile**

OUTPUT; CHAR(10) The user profile to use for this session. This parameter must be left justified and padded with blanks.

**Password**

OUTPUT; CHAR(10) The password to use for this session. This parameter must be left justified and padded with blanks.

**Initial current library**

OUTPUT; CHAR(10) The initial current library to be established for this session. This parameter must be left justified and padded with blanks.

**Related reference**

“TCPL0200 exit point format” on page 118

“TCPL0300 exit point format” on page 120

*TCPL0100 format usage notes:*

For File Transfer Protocol (FTP), if any of the returned output parameters are not valid, the FTP server will not allow the operation. In this case, the FTP server issues the message Data from exit program for exit point &1 is missing or not valid to the job log.

For FTP, if you encounter any exception when you call the exit program, the FTP server issues this message: Exception encountered for FTP exit program &1 in library &2 for exit point &3

This table summarizes what the FTP sever will do, depending on the value of the return code (parameter 8) that is returned to the FTP server by the exit program.

**Note:** A value of 'Return value' indicates that the exit program must return the appropriate value for that output parameter. The value will then be used by the FTP server to complete the logon request process.

Return code	User profile (9)	Password (10)	Initial lib (11)
0	Ignored	Ignored	Ignored
1	(User identifier, parameter 2)	(Password, parameter 4)	(From user profile)
2	(User identifier, parameter 2)	(Password, parameter 4)	Return value
3	Return value	Return value	(From user profile)
4	Return value	Return value	Return value
5	Return value	Ignored	(From user profile)
6	Return value	Ignored	Return value

In the table above, the values in parentheses indicate what the TCP/IP application uses for information when it ignores the output value. The entry Ignored means that it used no value; therefore return nothing for that return code value.

For the FTP server (exit point QIBM\_QTMF\_SVR\_LOGON, application identifier 1): when the user identifier is ANONYMOUS and this exit point adds the exit program, the server issues this special reply when requesting the password: 331 Guest logon in process, send complete e-mail address as password. The application issues this message before calling the exit program.

After the application accepts the server logon, the FTP server issues this reply: 230 Guest logon accepted, access restrictions apply

### REXEC server (application identifier 2):

1. If the return allow operation output parameter is not valid, the REXEC server will not allow the operation. The REXEC server issues the message "Data from exit program for exit point &1 is missing or not valid" to the job log.
2. If the REXEC server encounters any exception when calling the exit program, the REXEC server will not allow the operation. It issues the message "Exception encountered for REXEC exit program &1 in library &2 for exit point &3," to the job log.

### TCPL0200 exit point format:

Exit Point Format Name: TCPL0200

Exit Point Name: QIBM\_QTMF\_SVR\_LOGON

This is the required parameter group.

Parameter	Description	Input or output	Type and length
1	Application identifier	Input	Binary(4)
2	User identifier	Input	Char(*)
3	Length of user identifier	Input	Binary(4)
4	Authentication string	Input	Char(*)
5	Length of authentication string	Input	Binary(4)
6	Client IP address	Input	Char(*)
7	Length of client IP address	Input	Binary(4)
8	Allow logon	Output	Binary(4)
9	User profile	Output	Char(10)
10	Password	Output	Char(10)
11	Initial current library	Input/Output	Char(10)
12	Initial home directory	Output	Char(*)
13	Length of initial home directory	Input/Output	Binary(4)
14	Application-specific information	Input/Output	Char(*)
15	Length of application-specific information	Input	Binary(4)

## Parameter descriptions

### Application identifier

INPUT; BINARY(4) Identifies the application server from which the request is being made. The valid values are:

- 1 FTP server program

### User identifier

INPUT; CHAR(\*) The user identification supplied by the client program. For the FTP server, this parameter contains the data field from the USER subcommand.

**Length of user identifier**

INPUT; BINARY(4) The length (in bytes) of the user identifier string.

**Authentication string**

INPUT; CHAR(\*) The string (such as a password) supplied by the client program.

For the FTP server, this parameter contains the data field from the PASS (password) subcommand. Beginning with V5R1, if the user is authenticated by a client certificate, no data is provided for this parameter.

**Length of authentication string**

INPUT; BINARY(4) The length (in bytes) of the authentication string.

**Note:** For the FTP server: When the user is authenticated by a client certificate, this parameter is set to 0.

**Client IP address**

INPUT; CHAR(\*) The Internet Protocol (IP) address from which the session originates. This string is in dotted decimal format, left justified.

**Length of client IP address**

INPUT; BINARY(4) Indicates the length (in bytes) of the client IP address.

**Allow logon**

OUTPUT; BINARY(4) Indicates whether the logon operation should be accepted or rejected, and how password authentication is performed. The valid values are:

- 0 Reject the logon operation. Ignores all other output parameters.
- 1 Continue the logon operation with the specified user identifier and authentication string. The user identifier is the user profile, and the authentication string is the password. The current library and working directory is based on the settings of those output parameters. The application ignores the user profile and password output parameters.

**Note:** For the logon to succeed, the authentication string must match the user profile-specified password.

- 2 Continue the logon operation. Override the user profile and password with the returned values in the output parameters of this exit program. The application initializes the current library and working directory based on the settings of those output parameters.

**Note:** For the logon to succeed, the password output parameter must match the user profile-specified password.

**Attention!** IBM strongly recommends that you **never** code passwords directly in an exit program. Encryption, for example, allows algorithmic password determination.

- 3 Accept the logon operation. Override the user profile with the profile returned in the user profile output parameter of this exit program. The program initializes the current library and working directory based on the settings of the output parameters. It ignores the password output parameter.

**Note:** If the system is running at a security level of 20 or higher, specifying this value overrides normal i5/OS password processing. This is the only password authentication.

**User profile**

OUTPUT; CHAR(10) The user profile to use for this session. When required, this parameter must be left justified and padded with blanks.

**Password**

OUTPUT; CHAR(10) The password to use for this session. When required, this parameter must be left justified and padded with blanks.

**Initial current library**

OUTPUT; CHAR(10) The initial current library to use for this session. When required, this parameter must be left justified and padded with blanks. This parameter is set to the following special value when the exit program is called:

**\*CURLIB**

Use the current library that the user profile specifies.

**Initial home directory**

OUTPUT; CHAR(\*) The initial setting of the home directory to use for this session. When specified, this parameter must be a valid absolute path name, and the length of initial home directory parameter set to the proper value.

**Length of initial home directory**

INPUT/OUTPUT; BINARY(4) The length of the initial home directory parameter returned by the exit program. This parameter initializes at zero when the application calls the exit program. If the exit program does not change the value of the parameter, the home directory is initialized to the home directory that the user's profile specifies.

**Application-specific information**

INPUT/OUTPUT; CHAR(\*) Information that is used to communicate application-specific logon settings. For the correct format, see Format of application-specific information parameter.

**Length of application-specific information**

INPUT; BINARY(4) The length (in bytes) of the application-specific information.

**Related tasks**

"Configure FTP servers for graphical FTP clients and Web tools" on page 8

You can configure an File Transfer Protocol (FTP) server on the iSeries to support graphical FTP clients, Web browsers, and other Web tools.

**Related reference**

"TCPL0100 exit point format" on page 115

**TCPL0300 exit point format:**

Exit Point Format Name: TCPL0300

Exit Point Name: QIBM\_QTMF\_SVR\_LOGON

Exit Point Name: QIBM\_QTMX\_SVR\_LOGON

This is the required parameter group.

Parameter	Description	Input or output	Type and length
1	Application identifier	Input	Binary(4)
2	User identifier	Input	Char(*)
3	Length of user identifier	Input	Binary(4)
4	Authentication string	Input	Char(*)
5	Length of authentication string	Input	Binary(4)
6	CCSID of authentication string	Input	Binary(4)
7	Client IP address	Input	Char(*)
8	Length of client IP address	Input	Binary(4)

Parameter	Description	Input or output	Type and length
9	Allow logon	Output	Binary(4)
10	User profile	Output	Char(10)
11	Password	Output	Char(*)
12	Length of password	Output	Binary(4)
13	CCSID of password	Output	Binary(4)
14	Initial current library	Input/Output	Char(10)
15	Initial home directory	Output	Char(*)
16	Length of initial home directory	Input/Output	Binary(4)
17	CCSID of initial home directory	Input/Output	Binary(4)
18	Application-specific information	Input/Output	Char(*)
19	Length of application-specific information	Input	Binary(4)

## Parameter descriptions

### Application identifier

INPUT; BINARY(4) Identifies the application server from which the request is being made. The valid values are:

- 1 FTP server program
- 2 REXEC server program

### User identifier

INPUT; CHAR(\*) The user identification supplied by the client program.

For the FTP server, this parameter contains the data field from the USER subcommand.

### Length of user identifier

INPUT; BINARY(4) The length (in bytes) of the user identifier string.

### Authentication string

INPUT; CHAR(\*) The string (such as a password) supplied by the client program.

For the FTP server, this parameter contains the data field from the PASS (password) subcommand (unless the user is authenticated by a client certificate, in which case the client certificate is provided for this parameter).

### Length of authentication string

INPUT; BINARY(4) The length (in bytes) of the authentication string.

### CCSID of authentication string

INPUT; BINARY(4) The CCSID of the authentication string parameter. For the FTP server: When the user is authenticated by a client certificate, this parameter is set to -2.

### Client IP address

INPUT; CHAR(\*) The Internet Protocol (IP) address from which the session originates. This string is in dotted decimal format, left justified.

### Length of client IP address

INPUT; BINARY(4) Indicates the length (in bytes) of the client IP address.



### Allow logon

OUTPUT; BINARY(4) Indicates whether the logon operation should be accepted or rejected, and how password authentication is performed. The valid values are:

- 0 Reject the logon operation. Ignores all other output parameters.
- 1 Continue the logon operation with the specified user identifier and authentication string. The user identifier is the user profile, and the authentication string is the password. The current library and working directory is based on the settings of those output parameters. The application ignores the user profile and password output parameters.

**Note:** For the logon to succeed, the authentication string must match the user profile-specified password.

- 2 Continue the logon operation. Override the user profile and password with the returned values in the output parameters of this exit program. The application initializes the current library and working directory based on the settings of those output parameters.

**Note:** For the logon to succeed, the password output parameter must match the user profile-specified password.

*Attention!* IBM strongly recommends that you **never** code passwords directly in an exit program. Encryption, for example, allows algorithmic password determination.

- 3 Accept the logon operation. Override the user profile with the profile returned in the user profile output parameter of this exit program. The program initializes the current library and working directory based on the settings of the output parameters. It ignores the password output parameter.

**Note:** If the system is running at a security level of 20 or higher, specifying this value overrides normal i5/OS password processing. This is the only password authentication.

### User profile

OUTPUT; CHAR(10) The user profile to use for this session. When required, this parameter must be left-aligned and padded with blanks.

### Password

OUTPUT; CHAR(\*) The password to use for this session. When required, the Length of password and CCSID of password parameters must also be specified, and this parameter must be left-justified. When the QPWDLVL system value is set to 0 or 1, up to 10 characters can be specified; when the QPWDLVL system value is set to 2 or 3, up to 128 characters can be specified.

### Length of password

OUTPUT; BINARY(4) The length (in bytes) of the password. When required, the valid range is 1 to 512 bytes.

### CCSID of password

OUTPUT; BINARY(4) The CCSID of the password. This parameter must be set by the exit program when the password parameter is specified. The valid values are:

- 0 The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.

1-65533

A valid CCSID in this range.

### Initial current library

OUTPUT; CHAR(10) The initial current library to use for this session. When required, this



parameter must be left-aligned and padded with blanks. This parameter is set to the following special value when the exit program is called: \*CURLIB- Use the current library that the user profile specifies.

#### **Initial home directory**

OUTPUT; CHAR(\*) The initial setting of the home directory to use for this session. When specified, this parameter must be a valid absolute path name, and the length of initial home directory and CCSID of initial home directory parameters set to the proper values.

#### **Length of initial home directory**

INPUT/OUTPUT; BINARY(4) The length of the initial home directory parameter returned by the exit program. This parameter initializes at zero when the application calls the exit program. If the exit program does not change the value of the parameter, the home directory is initialized to the home directory that the user's profile specifies.

#### **CCSID of initial home directory**

OUTPUT; BINARY(4) The CCSID of the initial home directory. This parameter must be set by the exit program when the initial home directory is specified. The valid values are:

**0** The CCSID of the job is used to determine the CCSID of the data to be converted. If the job CCSID is 65535, the CCSID from the default CCSID (DFTCCSID) job attribute is used.

**1-65533**

A valid CCSID in this range.

#### **Application-specific information**

INPUT/OUTPUT; CHAR(\*) Information that is used to communicate application-specific logon settings. For the correct format, see Format of application-specific information parameter.

#### **Length of application-specific information**

INPUT; BINARY(4) The length (in bytes) of the application-specific information.

#### **Related tasks**

"Configure FTP servers for graphical FTP clients and Web tools" on page 8

You can configure an File Transfer Protocol (FTP) server on the iSeries to support graphical FTP clients, Web browsers, and other Web tools.

#### **Related reference**

"TCPL0100 exit point format" on page 115

## **Remove exit programs**

You can remove an installed exit program.

To remove an installed exit program, follow these steps:

1. Enter WRKREGINF at an iSeries command line.
2. Page down to an FTP Server Logon exit point:

```
QIBM_QTMF_SERVER_REQ  VLRQ0100
QIBM_QTMF_SVR_LOGON   TCPL0100
QIBM_QTMF_SVR_LOGON   TCPL0200
QIBM_QTMF_SVR_LOGON   TCPL0300
```
3. Enter 8 in the Opt field to the left of the exit point entry and press Enter.
4. At the **Work with Exit Program** display, enter a 4 (Remove).
5. Enter the name of the exit program in **Exit Program** field.
6. Enter the name of the library that contains the exit program in the Library field.
7. Press Enter.
8. After you finish removing exit points, stop and restart the FTP server.

#### **Related concepts**

“Manage access using File Transfer Protocol exit programs” on page 22

You can use File Transfer Protocol (FTP) exit points to protect the iSeries from this topic.

#### **Related tasks**

“Install and register exit programs” on page 14

You can create a library to contain your exit programs and their log files, compile the programs, and register them for use by the File Transfer Protocol (FTP) server.

“Start and stop the File Transfer Protocol server” on page 24

You can start and stop the File Transfer Protocol (FTP) server by using iSeries Navigator.

## **Data transfer methods**

Before you begin to transfer files, you must choose the appropriate file transfer type.

You can use the default type, ASCII, or specify a different type. ASCII is the Internet standard for character encoding. EBCDIC is the standard for iSeries. Select the appropriate type according to the following description:

- Use ASCII for transfers of files that only contain text (“text-only” files).
- Use EBCDIC to transfer EBCDIC data between systems that both support EBCDIC. This will avoid the need to convert data between EBCDIC and ASCII on both systems.
- Use BINARY for transfers of non-text files, such as binary numeric data, graphics files, and iSeries save files.

After you have chosen a file transfer format, you are ready to Transfer a file with FTP.

#### **Related tasks**

“Transfer files with File Transfer Protocol” on page 29

You can send and receive files with File Transfer Protocol (FTP).

#### **Related reference**

“ASCII (Change File Type to ASCII)” on page 60

“EBCDIC (Change File Type to EBCDIC)” on page 65

“BINARY (Set Transfer Type to Image)” on page 61

## **Transfer files that contain packed decimal data between iSeries servers**

Using File Transfer Protocol (FTP), you cannot convert special numeric formats like packed decimal or zoned decimal.

The transfer of packed decimal or zoned decimal data is supported between iSeries servers when you use either a transfer type of TYPE I (BINARY) or TYPE E (EBCDIC) with a transmission mode of BLOCK; these transfer types send the data as is without any conversion. The results of any other transfer type are unpredictable.

When transferring packed or zoned data in an externally-described QSYS.LIB file, the target file should be pre-created in the same manner as the source file. This restriction applies to data containing any special numeric format or when keyed access is required.

When transferring data with a transfer type of binary, the record length of the target file must be the same as the record length of the source file.

Before packed decimal or zoned decimal data can be transferred to or from other system architectures (such as S/390® or UNIX), you must convert the data to printable form.

## **Transfer \*SAVF files**

Because \*SAVF files must be sent as images, you must issue FTP BINARY subcommand before transferring these types of files.

When transferring a \*SAVF file using name format 0, the save file on the receiving system must be pre-created. It is recommended that files are pre-created in other situations as well for reasons of performance and integrity.

The transfer of a save file--because it is a file format peculiar to iSeries--can only be made usable if the sending and receiving servers are both iSeries servers. However, a save file can be sent to a non-iSeries server and stored there for backup purposes. The save file can be transferred later to the iSeries with FTP.

### **Example: Transferring a \*SAVF file from VM to an iSeries**

The following example shows how to transfer a \*SAVF file from VM to an iSeries for both NAMEFMT 0 and 1. The FTP session has already been initiated, the BINARY subcommand has been issued, and NAMEFMT 0 has been specified.

First, transfer the file P162484 SAVF310L from the VM A disk to the iSeries. VM FTP requires that you insert a period between its file name and file type. Give it the file name P162484 in library P162484 on the iSeries, and specify REPLACE as it has been pre-created even if it has not been used before. You will recall that pre-creation is mandatory with NAMEFMT 0.

Change the NAMEFMT to 1, and repeat the file transfer using the new name format. Once again, specify REPLACE, because the file exists from the previous step.

#### **Notes:**

- If you had not pre-created the file on the iSeries before performing the transfer with NAMEFMT 0, the transfer would have appeared to have completed satisfactorily. However, on inspection of the file on the iSeries, it would be seen that a physical file (\*PF) has been created and not a save file (\*SAVF).
- Some preprocessing might be necessary on the VM system depending on how the \*SAVF file was sent to VM:
  - If FTP was used to send the \*SAVF file to VM, you can just issue a GET subcommand to transfer it back to the iSeries.
  - If the Send Network File (SNDNETF) command was used to send the \*SAVF file to VM, it is first necessary to convert the file on the VM system from a record format (RECFM) of variable to a RECFM of fixed before using FTP to transfer it back to the iSeries server. To do this, use the COPYFILE command on VM. For example:

```
COPYFILE P162484 SAVF310L A = = = (RECFM F REPLACE
```

```

> GET P162484.SAVF310L P162484/P162484 (REPLACE
200 Port request OK.
150 Sending file 'P162484.SAVF310L'
250 Transfer completed successfully.
384912 bytes transferred in 3.625 seconds. Transfer rate106.183 KB/sec

> namefmt 1
202 SITE not necessary; you may proceed
Client NAMEFMT is 1.
> GET P162484.SAVF310L/QSYS.LIB/P162484.LIB/P162484.savf(REPLACE
200 Port request OK.
150 Sending file 'P162484.SAVF310L'
250 Transfer completed successfully.
384912 bytes transferred in 3.569 seconds. Transfer rate107.839 KB/sec
Enter an FTP subcommand.
====>

```

Figure 9. Transferring a \*SAVF from VM to iSeries using NAMEFMT 0 and NAMEFMT 1

## Transfer QDLS documents

When a QDLS document is transferred, the QDLS directory entry attribute that indicates the type of document is defaulted to the document type PCFILE on the receiving iSeries server for all document types except revisable-form text (RFT) documents.

RFT documents are defaulted to the document type RFTDCA. RFTDCA type documents can be viewed and edited using the WRKDOC CL command. PCFILE type documents cannot be viewed or edited using the WRKDOC CL command.

## Transfer root, QOpenSys, QLANSrv, QDLS, and QOPT files

You must use stream mode (MODE S) and file structure (STRUCT F) when transferring files in the root, QOpenSys, QLANSrv, QDLS, and QOPT file systems.

Root, QOpenSys, QDLS, and QOPT files can exist in any valid code page. Files transferred to the QLANSrv file system are tagged with the code page defined for the network server description corresponding to the directory containing that file.

Data conversion and CCSID assignments vary depending on the transfer TYPE used. You might want to refer to CCSID code page tagging for iSeries files. TYPE E is not supported for the QLANSrv file system.

When appending data to an existing file, the CCSID tag of that file is not changed. When appending data to an existing file using TYPE A, the data is converted to the code page of that file.

### Related reference

“CCSID code page tagging for iSeries files” on page 131

When File Transfer Protocol (FTP) creates a new file on an iSeries server, it is tagged with a Coded Character Set Identifier (CCSID) or the code page of that CCSID to identify the character data in that file. When replacing or appending data to an existing file, the tag of the file is not changed.

## Transfer files using QfileSvr.400

The QFileSvr.400 file system provides access to other file systems on remote iSeries servers.

The transfer of files in the “root”, QOpenSys, QLANSrv, QDLS, and QOPT file systems is supported. The transfer of files in the QSYS.LIB file systems is not supported.

You must use stream mode (MODE S) and file structure (STRUCT F). For example, in Figure 10, FILE.ABC is transferred to and from three different files systems on system AS012 using the QFileSvr.400 file system on system AS009.

After connecting to system AS009, the FTP client subcommands shown in Figure 11 perform the data transfers.

**Note:** The user ID and password on systems AS009 and AS012 must be the same.

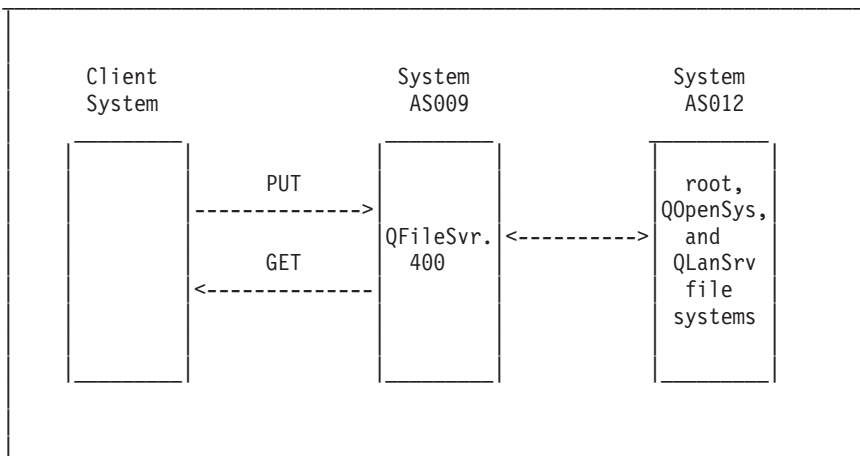


Figure 10. QFileSvr.400 file system example

```

NAMEFMT 1
LCD /CLIENTDIR1
CD /QFileSvr.400/AS012/FLSDIR
PUT FILE.ABC
GET FILE.ABC /CLIENTDIR2/FILE.ABC
CD /QFileSvr.400/AS012/QOpenSys/FLSDIR
PUT FILE.ABC
GET FILE.ABC /CLIENTDIR2/FILE.ABC (REPLACE
CD /QFileSvr.400/AS012/QLANSrv/NWS/LANSRV/DSK/K/FLSDIR
PUT FILE.ABC
GET FILE.ABC /CLIENTDIR2/FILE.ABC (REPLACE
SYSCMD RMVLNK '/CLIENTDIR2/FILE.ABC'
DELETE /QFileSvr.400/AS012/FLSDIR/FILE.ABC
DELETE /QFileSvr.400/AS012/QOpenSys/FLSDIR/FILE.ABC
DELETE /QFileSvr.400/AS012/QLANSrv/NWS/LANSRV/DSK/K/FLSDIR/FILE.ABC
QUIT

```

Figure 11. Subcommands to transfer files using QFileSvr.400

### Transfer QSYS.LIB files

This topic discusses the FTP operations for the QSYS.LIB file system.

Table 1 on page 128 and Table 2 on page 129 summarize FTP operations in stream transfer mode and in image transfer type for the QSYS.LIB file system. Keep the following points in mind when using these tables:

## Compatible record length and file size

When you send data to a file that already exists, the record and file size of the receiving file must be compatible with the file being sent or a transfer error will occur. Both the record and file size of the receiving file must be greater than or equal to the source file record and file size. To determine if the existing file size is compatible, you need to consider the current number of records, the number of extensions allowed, and the maximum record size allowed. You can view this information by entering the iSeries Display File Description (DSPFD) command.

## Automatic file creation on the iSeries server

When receiving a file, the iSeries server automatically creates a physical file, if one does not already exist. However, it is recommended that you pre-create the file on the iSeries.

## Data type

When transferring data using TYPE I, the data is not converted. If the file does not exist, it is tagged with CCSID 65535 when it is created.

**Note:** File pre-creation is advised when using the MGET and MPUT subcommands to transfer files with multiple members. When a file is not pre-created, FTP creates a file with a maximum record length equal to the longest record of the first pre-member processed. If the record length of any other file member is longer, a data truncation error will occur when transferring that member. Pre-creating a file with a record size to accommodate all members will prevent this error.

Table 1. Stream transfer mode for QSYS.LIB file system

Library exists	File exists	Member exists	Replace selected	Compatible record length	Compatible file size	Result
Yes	Yes	Yes	Yes	Yes	Yes	Data written to member.
Yes	Yes	Yes	No	N/A	N/A	Transfer rejected and message sent.
Yes	Yes	No	N/A	No	Yes	File transfer completed, records truncated, and message returned.
Yes	Yes	No	Yes	No	Yes	File transfer completed, records truncated, and message returned.
Yes	Yes	No	N/A	Yes	Yes	Member created and data written to it.
Yes	Yes	No	No	N/A	No	Transfer rejected and message sent.

Table 1. Stream transfer mode for QSYS.LIB file system (continued)

Library exists	File exists	Member exists	Replace selected	Compatible record length	Compatible file size	Result
Yes	No	N/A	N/A	N/A	N/A	File created with record length equal to the maximum record length of the incoming file. Member created and data written to member.
No	N/A	N/A	N/A	N/A	N/A	Transfer rejected and message sent. Use the CRTLIB command to create a library on the remote iSeries server.

Table 2. Image transfer type for QSYS.LIB file system

Library exists	File exists	Member exists	Replace selected	Result
Yes	Yes	Yes	Yes	Data written to member.
Yes	Yes	Yes	No	Transfer rejected and message sent.
Yes	Yes	No	N/A	Member created and data
Yes	No	N/A	N/A	
No	N/A	N/A	N/A	

### Related reference

“File pre-creation considerations” on page 130

It is strongly recommended that you pre-create any files that are to be transferred into the iSeries QSYS.LIB file system. This is the best method of ensuring that your data is transferred reliably and effectively with optimal performance and integrity.

### Receive text files to QSYS.LIB:

Because the iSeries QSYS.LIB file system internally supports a record structure, the iSeries File Transfer Protocol (FTP) converts files received on the iSeries server into a record structure and converts files sent from the iSeries server into the FTP file structure.

Text files received on the iSeries server by FTP are converted into a record structure in the following manner:

- When FTP receives a file and that file already exists on the iSeries server, the record length of the existing file is used.
- When FTP creates a new file on the iSeries server, it uses the length (excluding trailing spaces) of the longest line or record in the file as the record length of the file.

Text files sent from the iSeries server by FTP are converted into a file structure by removing the trailing blanks from each line or record and sending the truncated record.

### **File pre-creation considerations**

It is strongly recommended that you pre-create any files that are to be transferred into the iSeries QSYS.LIB file system. This is the best method of ensuring that your data is transferred reliably and effectively with optimal performance and integrity.

Be sure to allocate enough records to accommodate the entire file. On the iSeries this is done in the SIZE parameter of the Create Physical File (CRTPF) command.

Ensure that the RCDLEN parameter of the Create Physical File (CRTPF) command is adequate to accommodate the maximum record length expected.

**Note:** You can pre-create files on the FTP server system using the QUOTE subcommand. You can pre-create files on the FTP client system using the SYSCMD subcommand.

#### **Related reference**

“Transfer QSYS.LIB files” on page 127

This topic discusses the FTP operations for the QSYS.LIB file system.

### **Coded Character Set Identifier conversions**

iSeries uses Coded Character Set Identifier (CCSID) information to interpret the input data and provide the output data in the proper format for display.

The input might be ASCII or EBCDIC. The following topics provide detailed information about CCSID conversions:

#### **Specify mapping tables:**

For File Transfer Protocol (FTP) client, the ASCII mapping tables are specified in the FTP command. For FTP server this is done in the Change FTP Attributes (CHGFTP) command.

To specify the FTP client mapping tables:

1. Enter the command FTP.
2. Press F4. The **Start TCP/IP FTP** display is shown.
3. Press F10. The prompts for outgoing and incoming ASCII/EBCDIC tables are displayed.



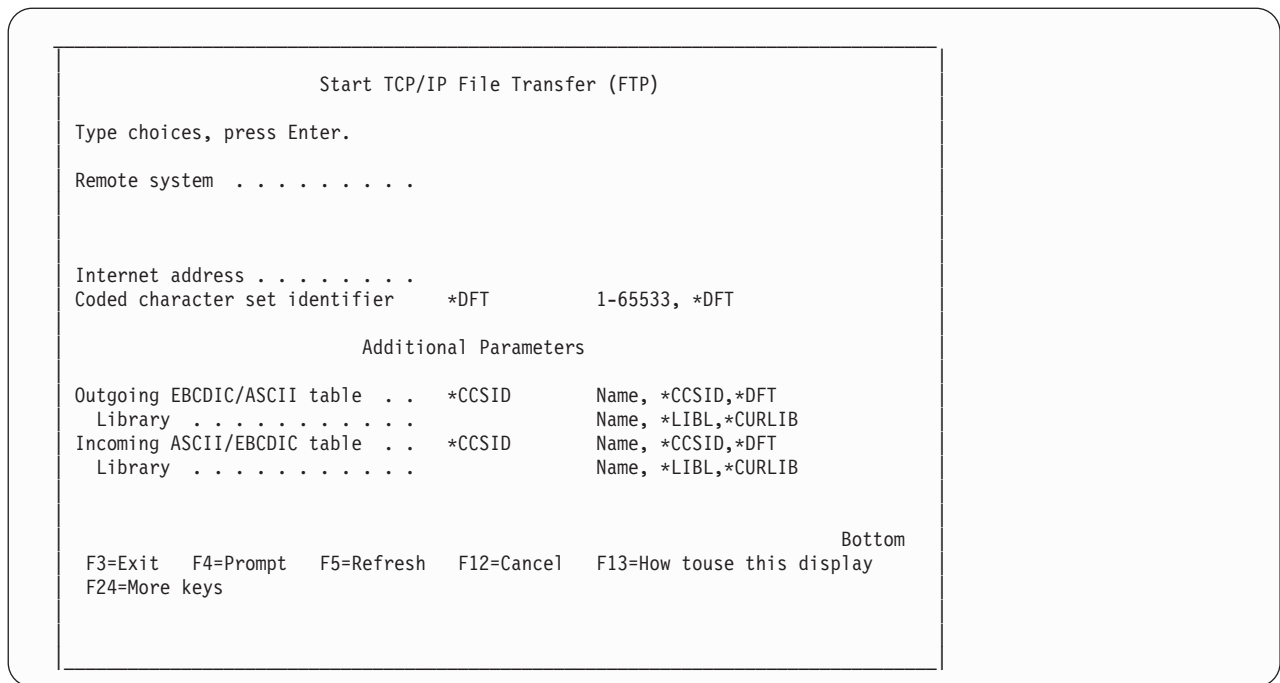


Figure 12. Specifying ASCII mapping tables with the \*CCSID value

Specify the CCSID (and hence the mapping tables) to be used for the FTP client. When the \*DFT value is not changed, the CCSID value 00819 (ISO 8859-1 8 bit ASCII) is used. You might also specify a specific CCSID for both inbound and outbound transfers. The use of CCSIDs is discussed in National Language Support considerations for FTP.

**Notes:**

- Double-byte character set (DBCS) CCSID values are not permitted for the CCSID parameter on the CHGFTP command. The DBCS CCSID values can be specified using the TYPE (Specify File Transfer Type) subcommand.
- IBM includes mapping support in FTP to ensure compatibility with releases prior to V3R1. Use of mapping tables for incoming TYPE A file transfers results in the loss of CCSID tagging if the target file must be created. IBM strongly recommends that you use CCSID support for normal operations.

**Related reference**

“National Language Support considerations for FTP” on page 132

This topic provides several points that you need to be aware of when using File Transfer Protocol (FTP) in an environment with different primary languages.

“TYPE (Specify File Transfer Type)” on page 86

**CCSID code page tagging for iSeries files:**

When File Transfer Protocol (FTP) creates a new file on an iSeries server, it is tagged with a Coded Character Set Identifier (CCSID) or the code page of that CCSID to identify the character data in that file. When replacing or appending data to an existing file, the tag of the file is not changed.

The following table summarizes how FTP assigns these values for different file systems and transfer types.

Table 3. CCSID code page tagging for iSeries files

Receiving file system	Transfer type A (ASCII)	Transfer transfer type C ('ccsid')	Transfer type E (EBCDIC)	Transfer type I (Image/Binary)
QSYS.LIB	CCSID specified by the EBCDIC coded character set identifier for new database files (CRTCCSID) setting.	'ccsid' if EBCDIC CCSID. If ccid is ASCII, then related default EBCDIC CCSID.	65535	65535
"root", QOpenSys, QDLS, QOPT	Default ASCII CCSID.	'ccsid' value specified in TYPE C ccid# subcommand.	Job CCSID if it is not 65535. If Job CCSID is 65535, assign Default Job CCSID.	Default ASCII CCSID.
QLanSrv	ASCII code page of the network server description for file directory.	ASCII code page of the network server description for file directory.	Not supported.	ASCII code page of the network server description for file directory.

**Note:** The default ASCII CCSID is defined when the FTP job is started: For the client, the CCSID parameter of the STRTCPFTP (and FTP) command. For the server, the CCSID parameter of the FTP Configuration attributes which can be changed using the CHGFTP command. QFileSvr.400 file assignments depend on the file system receiving the file.

#### Related reference

"Transfer root, QOpenSys, QLANSrv, QDLS, and QOPT files" on page 126

You must use stream mode (MODE S) and file structure (STRUCT F) when transferring files in the root, QOpenSys, QLANSrv, QDLS, and QOPT file systems.

#### National Language Support considerations for FTP:

This topic provides several points that you need to be aware of when using File Transfer Protocol (FTP) in an environment with different primary languages.

- When data is transferred using TYPE E (or EBCDIC), the data is stored as is and therefore will be in the EBCDIC code page of the file that it came from. This can result in the stored file being tagged with an inappropriate CCSID value when the primary language of the two iSeries servers is different. For example, when data in code page 237 is sent using TYPE E to the QSYS.LIB file system on a machine where the file does not exist, the data is stored as is in a new file tagged with CCSID 65535. If the receiving file already exists, then the data will be received as is and tagged with the existing file CCSID which can not be 237.

To avoid incorrect CCSID tagging, you can use the TYPE C CCSID subcommand (for example, TYPE C 237) to specify the CCSID of the data being transferred. When a CCSID is specified on a transfer and the data is written to an existing file, the data is converted to the CCSID of the existing file. If no target file exists before the transfer, a file is created and tagged with the specified CCSID.

In the preceding example, if the target file does not exist, a file with a CCSID of 237 is created on the receiving system. When the target file already exists, the data is converted from CCSID 237 to the CCSID of the target file.

- When starting the FTP client, message TCP3C14: Unable to convert data from CCSID &1 to CCSID &2, may be displayed. This occurs if no character conversion is available between the EBCDIC CCSID specified by your job and the ASCII CCSID specified for the this FTP session.

You can change the ASCII CCSID by specifying a value for the coded character set identifier parameter of the STRTCPFTP CL command. CCSID 850, which contains the IBM Personal Computer Latin-1 coded character set, is an ASCII CCSID for which character conversions are available to all valid job CCSID values.

- When using FTP in ASCII mode between two EBCDIC systems, the data on the system sending the file is converted from its stored EBCDIC code page to ASCII, and then from ASCII to the EBCDIC code page of the receiving system. Typically this does not present a problem because the 7-bit ASCII code page used by the two systems is the same unless the EBCDIC characters on the sending system are not defined in the ASCII code page. Also, some characters in the ASCII code page might be mapped differently between the two different EBCDIC code pages. This might occur if some of the ASCII characters are variant (the character occupies a different hexadecimal code point in an EBCDIC code page). The variant character might be interpreted differently on the receiving system if the EBCDIC code page is different from that of the system sending the file.

#### **Related reference**

“Specify mapping tables” on page 130

For File Transfer Protocol (FTP) client, the ASCII mapping tables are specified in the FTP command.  
For FTP server this is done in the Change FTP Attributes (CHGFTP) command.

## **File systems and naming conventions**

The File Transfer Protocol (FTP) server arranges the information units of a file system in a multiple-level tree-like structure.

The i5/OS file systems that you can use with FTP vary depending on the release level of the operating system. File systems on i5/OS can use different terms for data and the hierarchical grouping of data.

### **Naming conventions**

Each i5/OS file system has its own set of rules for naming files. The format you use to name any file must adhere to the naming conventions of the file system in which it resides. Formats and examples of file names for FTP-enabled i5/OS file systems are described in the topic Integrated file system. The server can provide naming information for files on non-iSeries servers when you use the QUOTE HELP subcommand.

#### **FTP server NAMEFMT**

When an FTP server session is started, NAMEFMT is set to a value of 0. You can change the NAMEFMT value by using the SITE subcommand.

The server automatically switches from the default of NAMEFMT 0 to NAMEFMT 1 when the ‘first’ file or pathname parameter received in a subcommand either:

- Starts with a slash (/) or a tilde (~) character

or

- Is blank (except for the LIST and NLST subcommands)

Any subsequent server subcommands with a file or path name parameter will not affect the NAMEFMT value. In addition to changing the NAMEFMT, the server reply for the subcommand will include a statement saying that the NAMEFMT value has been changed.

For example, the server NAMEFMT value will be changed to “1” if the first server subcommand with a file or path name is:

```
CWD /DIR1/DIR2A
```

The server reply will be:

```
250-NAMEFMT set to 1.  
250 Current directory changed to /DIR1/DIR2A.
```

**Note:** This capability enables the typical Web browser, which requires NAMEFMT 1, to interact with iSeries FTP servers without issuing a SITE NAMEFMT 1 subcommand.

**Related concepts**

Integrated File System

Files and file systems

**Related reference**

“NAMEFMT (Select File Naming Format)” on page 73

“QUOTE (Send a Subcommand to an FTP Server)” on page 77

## **i5/OS file systems that are supported by File Transfer Protocol**

The file systems you can use with File Transfer Protocol (FTP) vary depending on the release level of the operating system.

**QSYS.LIB Library file system - libraries, files, members**

FTP supports the transfer of save files and members in physical files, logical files, DDM files, and source physical files. For QSYS.LIB file system physical files, the data transferred is a member of a file which resides in a library.

**QDLS Document library services - folders and documents**

For the Document Library Services (QDLS) file system the data transferred is a document. QDLS documents reside in directories called folders.

**“root”** The / file system. This file system takes full advantage of the stream file support and hierarchical directory structure of the integrated file system. It has the characteristics of the DOS and OS/2<sup>®</sup> file systems.

**QOpenSys**

The open systems file system. This file system is compatible with UNIX-based open system standards, such as POSIX and XPG. Like the root file system, it takes advantage of stream file and directory support that are provided by the integrated file system. It supports case-sensitive names.

**QOPT** The QOPT optical file system. This file system provides access to stream data that is stored on optical media.

**QFileSvr.400**

The i5/OS file server file system. This file system provides access to other file systems that reside on remote iSeries servers. With FTP you do not have access to QSY.LIB, QDLS, and QOPT that uses QFileSvr.400.

**Related concepts**

Integrated file system

## **File Transfer Protocol server reply status messages**

This topic contains access common reply codes and what they indicate.

When you enter subcommands during an File Transfer Protocol (FTP) client session, status messages return to your display in a 3-digit code: *xyz*.

The first digit (x) tells you whether the response is good, bad, or incomplete. There are five values for the first digit:

- 1yz = Good. The requested action is being initiated; another reply should follow.
- 2yz = Good. The requested action was successfully completed; a new request may be initiated.
- 3yz = Incomplete. The subcommand was accepted, but the requested action is being held pending receipt of more information.

- 4yz = Incomplete. The server did not accept the subcommand. The requested action did not take place; the error is temporary and you can request the action again.
- 5yz = Bad. The subcommand was not accepted, and the requested action did not take place.

The second digit (y) tells you the functional category of the response.

- x0z=Syntax. Refers to syntax errors, commands that aren't appropriate for what you're trying to do, and unnecessary commands.
- x1z=Information. Refers to requests for information, such as status or help.
- x2z=Connections. Refers to the control or data connections.
- x3z=Authentication. Refers to the login process.
- x5z=File system. Refers to the status of the server in relation to the file transfer request.

The third digit (z) tells you a finer level of detail about the functional category.

The following table describes the common reply codes and what they indicate. The message text might vary for different server systems.

Code	Meaning
110	Restart the marker reply
120	Service is ready in nnn minutes
125	Data connection is already open; transfer is starting
150	File starting OK; about to open the data connection
200	Command OK
202	Command was not implemented; it is not used on this system
211	System status, or system help reply
212	Directory status
213	File status
214	Help message
220	Service is ready for a new user
226	Closing the data connection; the requested file action was successful
230	User is logged in
250	Requested file action was okay; action is completed
257	Path name was created
331	Password is required
332	Account is required
425	Cannot open the data connection
426	Connection is closed; the transfer ended abnormally
450	Requested file action was not taken; file busy
451	Requested action ended abnormally; local error in processing
452	Requested action was not taken; insufficient storage exists in system
500	Syntax error; command was unrecognized
501	Syntax error in the parameters or arguments

Code	Meaning
502	Command was not implemented
503	Bad sequence of commands
504	Command was not implemented for that parameter
530	Logon attempt was rejected
532	Need an account for storing files
550	Requested action was not taken; the file was not found (or no access)
551	Requested action ended abnormally; the page type is unknown
552	Requested file action ended abnormally; storage allocation was exceeded
553	Requested action was not taken; the file name is not allowed

### Related reference

“File Transfer Protocol server subcommands” on page 39

These subcommands represent communication between the client and server. This topic includes descriptions for iSeries CL-equivalent subcommands that are unique to iSeries FTP server.

“File Transfer Protocol client subcommands” on page 57

Use File Transfer Protocol (FTP) client subcommands to establish a connection with a remote FTP server, navigate libraries and directories, create and delete files, and transfer files.

## File Transfer Protocol server syntax conventions

The File Transfer Protocol (FTP) server subcommands described in this topic make use of syntax conventions:

The following syntax are conventions used by FTP server subcommands:

### Uppercase Letters

You must enter letters in uppercase exactly as shown in the syntax definitions for subcommands.

You can enter these letters in either uppercase or lowercase.

### Lowercase Words or Hyphenated Terms

Lowercase words or hyphenated terms, such as (remotefile and account-information,) represent variables for which you must substitute specific information.

### Brackets [ ]

You can consider words, symbols, or phrases placed within brackets to be optional.

### Left Parentheses ( and Asterisks \*

You must enter left parentheses and asterisks exactly as shown in the syntax definitions.

### Braces { }

Braces indicate a group of parameters, values, or variables that you can repeat.

### Ellipsis ...

Ellipses indicate that you can include zero or more repetitions of the preceding variable enclosed within brackets.

### Vertical Bar |

A vertical bar between parameters or values indicates that you can specify one or the other, but not both, at one time. You can see the vertical bars placed within sets of brackets or braces.

### Related reference

“File Transfer Protocol server subcommands” on page 39

These subcommands represent communication between the client and server. This topic includes descriptions for iSeries CL-equivalent subcommands that are unique to iSeries FTP server.

“ADDM (Add Physical File Member)” on page 41

“ADDV (Add Physical File Variable Length Member)” on page 41

“APPE (Append to Existing File)” on page 42

“AUTH (Authorization)” on page 42

“CRTL (Create Library)” on page 43

“CRTP (Create Physical File)” on page 43

“CRTS (Create Source Physical File)” on page 44

“CWD (Change Working Directory or Library)” on page 44

“DEBUG (Turn on the FTP Server Trace)” on page 44

“DELE (Delete File or Document)” on page 45

“DLTF (Delete File)” on page 45

“DLTL (Delete Library)” on page 45

“HELP (Getting Help from an iSeries Remote Server)” on page 45

“LIST (File List)” on page 46

“MKD (Make Directory)” on page 46

“MODE (Set Transfer Mode)” on page 46

“NLST (Name List)” on page 47

“NOOP (Obtain Server Response)” on page 47

“PASS (Password)” on page 48

“PASV (Use Passive Data Connection)” on page 48

“PBSZ (Protection Buffer Size)” on page 48

“PORT (Data Port)” on page 48

“PROT (Data Channel Protection Level)” on page 49

“PWD (Display Working Directory or Library)” on page 49

“QUIT (End an FTP Server Session)” on page 50

“RCMD (Send a CL Command to an FTP Server System)” on page 50

“REIN (Reinitialize Session between Systems)” on page 51

“RETR (Retrieve File)” on page 51

“RMD (Remove Directory)” on page 51

“RNFR (Rename From)” on page 51

“RNTO (Rename To)” on page 52

“SITE (Send Information Used by a Server System)” on page 52

“STOR (Store File)” on page 54

“STOU (Store Unique)” on page 54

“STRU (Specify File Structure)” on page 55

“SYST (Identify the Name of the Operating System)” on page 55

“TIME (Set Time-Out Values for FTP Server)” on page 55

“TYPE (Specify Representation Type)” on page 56

“USER (Send a User Logon ID to the Server)” on page 57

## File Transfer Protocol client syntax conventions

The File Transfer Protocol (FTP) client subcommands described in this topic make use of syntax conventions:



The following syntax conventions are used by the FTP client subcommands:

### **Uppercase Letters**

Letters printed in uppercase in the syntax definitions for client subcommands are the minimum number of letters that you must enter. You can enter FTP client subcommands in either uppercase or lowercase.

### **Lowercase Words or Hyphenated Terms**

Lowercase words or hyphenated terms, like `remotefile` and `account-information`, represent variables that you must substitute specific information.

### **Brackets [ ]**

You can consider words, symbols, or phrases placed within brackets to be optional.

### **Left Parentheses ( and Asterisks \***

You must enter left parentheses and asterisks exactly as they appear in the syntax definitions.

### **Braces { }**

Braces indicate a group of parameters, values, or variables that you may repeat.

### **Ellipsis ...**

Ellipses indicate that you can include zero or more repetitions of the preceding variable enclosed within brackets.

### **Vertical Bar |**

A vertical bar between parameters or values indicates that you can specify one or the other, but not both, at one time. The vertical bars are within sets of brackets or braces.

#### **Related reference**

“File Transfer Protocol client subcommands” on page 57

Use File Transfer Protocol (FTP) client subcommands to establish a connection with a remote FTP server, navigate libraries and directories, create and delete files, and transfer files.

“ACCT (Send Account Information)” on page 59

“APPEND (Append a Local File Member to a Remote File)” on page 60

“ASCII (Change File Type to ASCII)” on page 60

“BINARY (Set Transfer Type to Image)” on page 61

“CD (Change Working Directory or Library)” on page 61

“CLOSE (End an FTP Session with the Remote System)” on page 62

“DEBUG (Create Client Trace and Control Display of Server Subcommands Sent to Remote System)” on page 62

“DEBUG (Change Client Time-Out Limit Values)” on page 63

“DELETE (Delete a File on a Remote System)” on page 64

“DIR (List Directory Entries, Libraries, or Files)” on page 64

“EBCDIC (Change File Type to EBCDIC)” on page 65

“GET (Copy a File from a Remote System to the Local System)” on page 65

“HELP (Getting Help for FTP Subcommands)” on page 66

“LCD (Change Working Library or Directory on Local System)” on page 67

“LOCSITE (Specify Local Site Information)” on page 68

“LOCSTAT (Display Local Status Information)” on page 69

“LS (List Remote File Names)” on page 69

“LTYPE (Local Type)” on page 70

“MDELETE (Delete Multiple Files on a Remote System)” on page 70

“MGET (Copy Multiple Files from a Remote System to the Local System)” on page 71

“MKDIR (Make Directory)” on page 72



"MODE (Specify Transmission Mode of Data)" on page 72  
 "MPUT (Send Multiple File Members from the Local System to a Remote System)" on page 73  
 "NAMEFMT (Select File Naming Format)" on page 73  
 "NULLFLDS (Allow Transfer of Files with NULL Fields)" on page 74  
 "OPEN (Connect to FTP Server on a Remote System)" on page 75  
 "PASS (Send Your Password)" on page 75  
 "PUT (Copy a File Member from the Local System to a File on a Remote System)" on page 75  
 "PWD (Display Current Directory, Folder, or Library)" on page 76  
 "QUOTE (Send a Subcommand to an FTP Server)" on page 77  
 "REINITIALIZE (Reinitialize Session between Systems)" on page 78  
 "RENAME (Rename a File on a Remote System)" on page 78  
 "RESET (Reset)" on page 79  
 "RMDIR (Remove Directory)" on page 79  
 "SECDATA (Setting Data Security Protection)" on page 79  
 "SECOPEN (Setting Data Security Protection)" on page 80  
 "SENDPASV (Specify Whether to send a PASV Subcommand)" on page 81  
 "SENDPORT (Specify Whether to Sends a PORT Subcommand)" on page 82  
 "SENDSITE (Specify Whether to Send a SITE Subcommand)" on page 83  
 "SITE (Send Information Used by a Remote System)" on page 83  
 "STATUS (Retrieve Status Information from a Remote System)" on page 83  
 "STRUCT (Specify File Structure)" on page 84  
 "SUNIQUE (Control Overwriting of Files)" on page 84  
 "SYSCMD (Pass an iSeries CL Command to Your Local iSeries)" on page 85  
 "TYPE (Specify File Transfer Type)" on page 86  
 "USER (Send Your User ID to the Remote System)" on page 87  
 "VERBOSE (Control of Text Display of Error Reply Messages)" on page 87

## Enclose subcommand parameters

You can use either a single quotation mark (') or quotation marks (") to enclose parameters from this topic.

You can use either a single quotation mark (') or quotation marks (") to enclose subcommand parameters. To enclose a single quotation mark within a parameter, you must enter it either as two consecutive single quotation marks (') in a parameter that is enclosed by a single quotation mark, or you must enter it as a single quotation mark in a parameter that is enclosed by quotation marks (").

Similarly, if a quotation mark (") is to be contained within a parameter, you must enter it in one of these ways:

- A quotation mark (") in a parameter that is enclosed by apostrophes.
- As two consecutive quotation marks (") in a parameter that is enclosed by quotation marks.

You can use the apostrophe or quotation marks as follows:

1. If the apostrophe or quotation marks within the parameter are the same as the starting and ending delimiter, you must repeat the mark within the parameter. For example:

```

'ABCD'12345'
  results in  ABCD'12345
"ABCD""12345"
  results in  ABCD"12345
  
```

2. If the starting and ending marks are not the same as the mark within the parameter, you do not repeat the mark. For example:

```
"ABCD'12345"
  results in ABCD'12345
'ABCD"12345'
  results in ABCD"12345
```

3. If both the apostrophe and quotation marks are within the parameter, you must choose one mark symbol as the delimiter. For example:

```
"ABC'12""345" or 'ABC'12"345'
  results in ABC'12"345
```

#### Related reference

“PUT (Copy a File Member from the Local System to a File on a Remote System)” on page 75

“File names for client-transfer subcommands”

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

“Naming files for transfer” on page 142

Use the `localfile` and `remotefile` parameters to transfer data.

### File names for client-transfer subcommands

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

The FTP client provides a default file name if the target file name for the PUT, APPEND, and GET subcommands is omitted. Because you can specify source file names for the MPUT and MGET subcommands, the FTP server also generates target file names for MPUT and MGET. See the following Data Transfer Subcommands table for the syntax of these subcommands. The table column labeled *Target* is the parameter for which a default name is provided.

Subcommand	Source	Target	Other
APPEND	local filename	[server filename]	
PUT	local filename	[server filename]	
GET	server filename	[local file name]	[(Replace)]
MPUT	local filename		
MGET	server filename		[(Replace)]

### PUT and APPEND subcommands

For the PUT and APPEND subcommands, the rules for forming default names are divided into two categories:

- iSeries server case
  - If the target file system is a library file system or a document library system, the default name complies with the naming rules for these systems, including their name format.
  - If the target file system is neither a library file system nor a document file system it is one of two names:
    - The default name is the name after the last slash in the source file name
    - The *same* as the source file name if there is no slash.
- Non-iSeries server case
  - If the source file is a library file system file, then the default name consists of the *file name.member name*. If there is no member name, the file name is the default name.
  - If the source file is a document library services file, the default name is the file name and the extension.

- If the source file is neither a library file system nor a document library services file, the name after the last slash in the source name is the default name. If there is no slash, the default name is the same as the source name.

If the server is an iSeries server, then the server generates the default name in these subcommands using the same rules as applied for the PUT subcommand.

## GET and MGET subcommand

If the server is not an iSeries server, it bases the default name for the GET and MGET subcommands on the part of the source name that follows the last slash. If there is no slash, the entire source name is the default name. Here are the rules for forming default names:

- If the client file system is the *library file system* (iSeries database), these rules apply:
  - If the remote file name contains a period (.), the characters preceding the period are truncated to 10 characters to form the local file name. The characters after the period are truncated to 10 characters to form the member name.
  - If the remote file name does not contain a period, both file and member names are set to the remote file name truncated to 10 characters to form the local file name.
  - If the name format is 1, the server adds the appropriate extensions to the file and member parts of the name.
- If the client file system is *document library services*, these rules apply:
  - If the remote name contains a period, the characters preceding the period are truncated to 8 characters. The characters after the period are truncated to 3 characters.
  - If the remote name does not contain a period, the name is truncated to 8 characters without an extension.
- For other file systems, the name after the last slash in the remote name is the default name.

### Notes:

1. Save files do not have members, so default names for save files do not have a member part.
2. The server displays the default names when the DEBUG mode is on.

### More details on syntax:

FTP client syntax conventions

## Naming files for transfer

The FTP client subcommands that you use for transferring data can have a **localfile** or a **remotefile** parameter or both. You can use these parameters to name the data you want to transfer. The transfer subcommands are:

```
APPEND localfile [remotefile]
DELETE remotefile
GET remotefile [localfile]
MDELETE remotefiles
MGET remotefiles
MPUT localfiles
PUT localfile [remotefile]
```

The names for the localfile and remotefile parameters can be either partially qualified or fully qualified. A partially-qualified name includes the name of the data itself as well as one or more names in the hierarchical sequence above the data. A fully-qualified name includes all names in the hierarchical sequence above the data.

When the name is partially qualified, the current working directory identifies the file to be processed. You can set the working directory on the local client system with the LCD subcommand. You can set the working directory on the remote server system with the CD subcommand.

The format of the localfile name parameters must conform to iSeries file naming rules. The remotefile names must adhere to the file naming rules of the remote system.

#### **More details on syntax:**

- Enclosing subcommand parameters: You can use either a single quotation mark (') or quotation marks (") to enclose parameters.
- FTP client syntax conventions: FTP client subcommands make use of these syntax conventions.

#### **Related tasks**

"Enclose subcommand parameters" on page 139

You can use either a single quotation mark (') or quotation marks (") to enclose parameters from this topic.

#### **Related reference**

"APPEND (Append a Local File Member to a Remote File)" on page 60

"GET (Copy a File from a Remote System to the Local System)" on page 65

"PUT (Copy a File Member from the Local System to a File on a Remote System)" on page 75

"MPUT (Send Multiple File Members from the Local System to a Remote System)" on page 73

"MGET (Copy Multiple Files from a Remote System to the Local System)" on page 71

"DEBUG (Change Client Time-Out Limit Values)" on page 63

"LCD (Change Working Library or Directory on Local System)" on page 67

"CD (Change Working Directory or Library)" on page 61

"DELETE (Delete a File on a Remote System)" on page 64

"MDELETE (Delete Multiple Files on a Remote System)" on page 70

"Naming files for transfer"

Use the localfile and remotefile parameters to transfer data.

## **Naming files for transfer**

Use the localfile and remotefile parameters to transfer data.

The FTP client subcommands that you use for transferring data can have a **localfile**, a **remotefile** parameter, or both. You can use these parameters to name the data you want to transfer. The transfer subcommands are:

APPEND localfile [remotefile]

DELETE remotefile

GET remotefile [localfile]

MDELETE remotefiles

MGET remotefiles

MPUT localfiles

PUT localfile [remotefile]

The names for the localfile and remotefile parameters can be either partially qualified or fully qualified. A partially-qualified name includes the name of the data itself as well as one or more names in the hierarchical sequence above the data. A fully-qualified name includes all names in the hierarchical sequence above the data.

When the name is partially qualified, the current working directory identifies the file to be processed. You can set the working directory on the local client system with the LCD subcommand. You can set the working directory on the remote server system with the CD subcommand.

The format of the localfile name parameters must conform to iSeries file naming rules. The remotefile names must adhere to the file naming rules of the remote system.

#### **More details on syntax:**

- Enclosing subcommand parameters: You can use either a single quotation mark (') or quotation marks (") to enclose parameters.
- Default file names for client transfer subcommands: [Link to this information about default file names for client transfer subcommands.](#)
- FTP client syntax conventions: FTP client subcommands make use of these syntax conventions.

#### **Related tasks**

[“Enclose subcommand parameters” on page 139](#)

You can use either a single quotation mark (') or quotation marks (") to enclose parameters from this topic.

#### **Related reference**

[“APPEND \(Append a Local File Member to a Remote File\)” on page 60](#)

[“DELETE \(Delete a File on a Remote System\)” on page 64](#)

[“GET \(Copy a File from a Remote System to the Local System\)” on page 65](#)

[“MDELETE \(Delete Multiple Files on a Remote System\)” on page 70](#)

[“MGET \(Copy Multiple Files from a Remote System to the Local System\)” on page 71](#)

[“MPUT \(Send Multiple File Members from the Local System to a Remote System\)” on page 73](#)

[“PUT \(Copy a File Member from the Local System to a File on a Remote System\)” on page 75](#)

[“LCD \(Change Working Library or Directory on Local System\)” on page 67](#)

[“CD \(Change Working Directory or Library\)” on page 61](#)

[“File names for client-transfer subcommands” on page 140](#)

You can use default source and target file names for some subcommands; however, you must specify a file name for other subcommands.

---

## **Troubleshoot FTP**

This topic provides basic troubleshooting techniques for problems that occur with the FTP server or FTP client.

### **Determine problems with File Transfer Protocol**

You can view a list of steps to determine whether your Simple Mail Transfer Protocol (SMTP) is working correctly.

If you detect a problem when using File Transfer Protocol (FTP), use the following flow chart to identify the cause after using the flow chart for general TCP/IP problems. The cause lists that follow list steps to help you identify the cause of the problem.

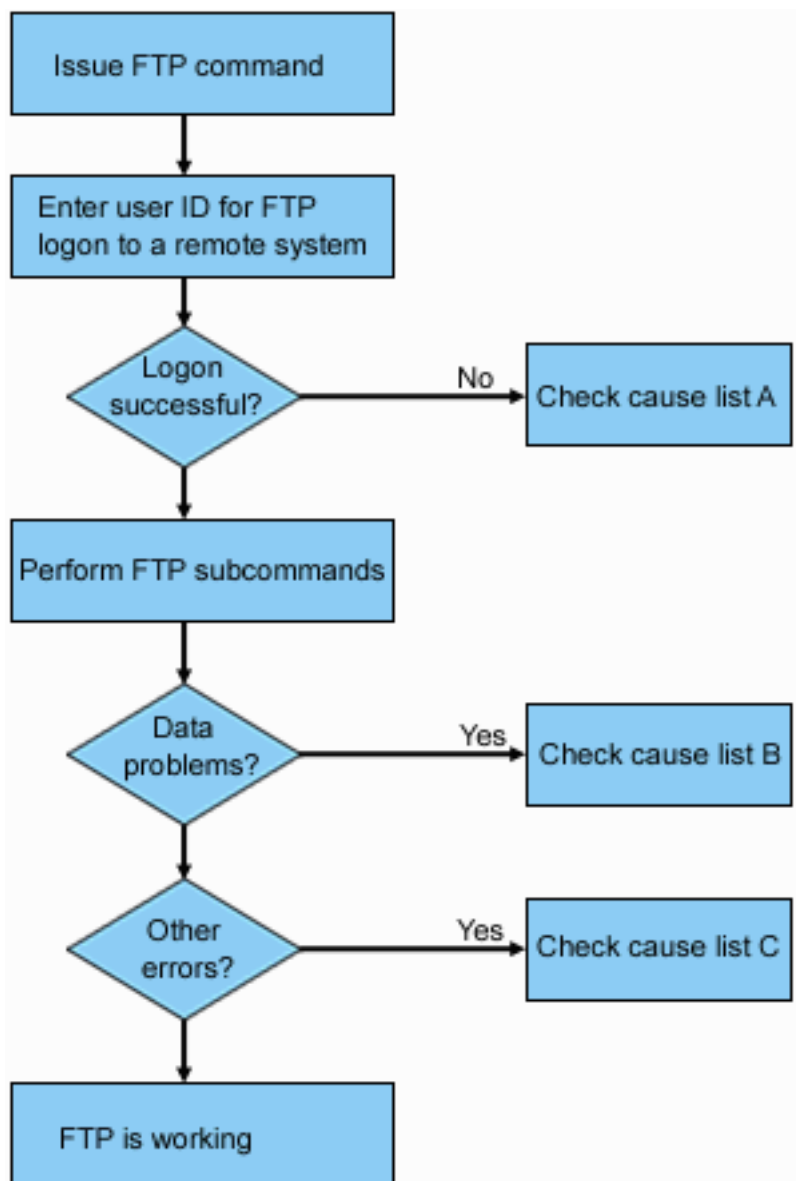


Figure 13. FTP problem analysis

### Cause list A

1. Is there is a long delay between connecting to the iSeries FTP server and receiving a prompt for a user id? If so, check the configuration of the domain name server on your iSeries. The FTP server performs a DNS query as soon as a new connection is received. DNS problems can cause the server to hang for several minutes before a response is received.
2. Check to see if an exit program has been added to the FTP Server Logon Exit Point. If yes, then check if the logon that is unsuccessful is allowed by the exit program.
3. Check to see if the remote logon requires a password if a password was requested. Some systems request a password, but the connection can fail because it is not required.
4. Set up a password on the remote system if required. You might need to restart if you change the security information about the system.
5. Check your user ID and password by attempting to sign on to your remote system. If you are unable to do so, contact the system owner to verify that your user ID and password are correct.

## Cause list B

1. Make sure binary mode is in effect if you are transferring binary files.
2. Check to be sure the mapping tables on both the client and server systems are compatible. You need only do this if you are using your own mapping tables.
3. Check to see that the correct CCSID has been specified for the transfer. If not, use the TYPE or LTYPE subcommand to set the correct CCSID value before the transfer is performed.
4. Create a file on the system that you are planning to store data into. Set the proper record length, number of members, and number of increments. Try the data transfer again and verify that it was successful.
5. Make sure that you are authorized to use the file and the file members.
6. Check to see if the transfer file contains packed decimal or zoned decimal data.
7. If you are transferring a Save file, verify that the appropriate method was used.

## Cause list C

1. Check file size limits on the remote system.
2. Check to see if the FTP server timer ended. The iSeries server time-out value can be set using the QUOTE TIME command.
3. Use the NETSTAT command to verify that the \*LOOPBACK interface is active. Then re-create the problem doing FTP LOOPBACK (iSeries-to-iSeries internally).
  - If the problem cannot be recreated, it is probably a remote system problem.
  - If you can re-create the problem, complete the following steps:
    - a. If the problem is an FTP server problem, then start the FTP server trace using the TRCTCPAPP command.
    - b. Create the problem again.
    - c. End the FTP connection. Refer to the Starting and stopping the FTP server.
    - d. End the FTP server trace using the TRCTCPAPP command.
    - e. Find a spooled file with the following characteristics:
      - The file name is QTMFFTRC
      - The username associated with the file is the name of the user who issued the TRCTCPAPP command.The trace is a spooled file in the default output queue of the system associated with the FTP server job.
    - f. Send in that spooled file.
    - g. If the problem was on the iSeries FTP client, a trace can be obtained using the DEBUG 100 client subcommand.
    - h. When running the FTP client interactively, use the F6 (Print) key to create a spool file that contains a history of the FTP client subcommands entered, and the associated FTP server replies. When the FTP client is run in batch unattended mode, then this history of subcommands and server replies is written to the specified OUTPUT file.

### Related tasks

“Start and stop the File Transfer Protocol server” on page 24

You can start and stop the File Transfer Protocol (FTP) server by using iSeries Navigator.

### Related reference

“Server logon exit point” on page 100

You can control the authentication of users to a TCP/IP application server with the TCP/IP Application Server Logon exit point.

## Materials required for reporting FTP problems

This topic describes the information that the IBM service representative might require to resolve an FTP problem.

Any FTP problem reported to IBM should include the following information:

- A communications trace from the time of the failure (Request TCP/IP data only) formatted twice: once for ASCII and once for EBCDIC.
- If the FTP client or server has logged software error data, submit the data.

**Note:** The system value QSFWERRLOG must be set to \*LOG for software error logging to take place. If an error occurs while QSFWERRLOG is set to \*NOLOG, change the value to \*LOG, try to re-create the error, and submit the logged software error data. If logged software error data is submitted, there is no need to perform a trace of FTP.

- The QTCPIP and any FTP server or FTP client job logs.
- The FTP client and FTP server debug traces.
- For FTP client problems, a spool file containing the FTP client session (which can be obtained by hitting the print (F6) key in the FTP session).
- If data integrity is the problem, then the file, member, or library causing the problem should be sent in along with a copy of the description of the file, member, or library.

### Related concepts

“Trace the FTP client” on page 149

You can use the DEBUG subcommand to track down problems on the FTP client.

“Trace the FTP server”

You can use the DBUG subcommand to track down problems on the FTP server.

## Trace the FTP server

You can use the DBUG subcommand to track down problems on the FTP server.

The FTP server can be traced from any iSeries or non-iSeries server that runs TCP/IP. There are two ways to trace the FTP server. The FTP server DBUG subcommand traces within an FTP server session. The Trace TCP/IP Application (TRCTCPAPP) command allows system wide tracing of all the FTP servers.

## Trace the FTP server with the DBUG subcommand

To trace the FTP server, follow these steps:

1. Type QUOTE DBUG to start the trace.



```

File Transfer Protocol

Previous FTP subcommands and messages:
Connecting to host name xxxxxnnn.xxxxxxxx.xxx.xxx at address
n.nnn.nn.nnn using port 21.
220-QTCP at xxxxxnnn.xxxxxxxx.xxx.xxx.
220 Connection will close if idle more than 5 minutes.
215 i5/OS is the remote operating system. The TCP/IPversion is
"V4R4M0".
>
331 Enter password.
230 TEST logged on.
250 Now using naming format "0".
257 "QGPL" is current library.
> quote dbug
250 Debug mode is now ON.
Enter an FTP subcommand.
====> quote dbug

F3=Exit      F6=Print      F9=Retrieve
F17=Top      F18=Bottom    F21=CL command line

```

2. Perform the FTP operation that you want to trace.
3. Type QUOTEDBUG again to end the trace. The trace creates a spooled file called QTMFFTRC. The default output queue contains the spooled file. The user is always the name of the user who was logged on to the FTP server when the trace was ended.
4. Type QUIT to end the FTP session.
5. Enter the following command to find the output queue.

DSPSYSVAL QPRTDEV

For example, the following display appears:

```

Display System Value
System value . . . . . : QPRTDEV
Description . . . . . : Printer device description
Printer device . . . . . : PRT01      Name

```

The printer device is also the name of the default system output queue.

6. Record the name of the printer device. In this example, PRT01 is the printer device.
7. Press F12 (Cancel) to return to the display where you entered the DSPSYSVAL command.
8. Type the following command:

WRKOUTQ OUTQ(printer-device)

Replace printer-device with the printer device recorded in the previous display. PRT01 is the output queue in this example. For example, the following display appears:

```

Work with Output Queue
Queue: PRT01      Library: QGPL      Status: RLS
Type options, press Enter.
  1=Send  2=Change  3=Hold  4=Delete  5=Display  6=Release  7=Messages
  8=Attributes  9=Work with printing status
Opt File      User      User Data  Sts  Pages  Copies  Form Type  Pty
-  QTCPPRT    QTCP      QTMSMTP    HLD   46    1      *STD      5
-  QTMFFTRC   QSECOFR      HLD   44    1      *STD      5

```

9. Press F18 (Bottom) to get to the bottom of the spooled file list if More...appears on the display.
10. Find the last file named QTMFFTRC with the same user as the user who was logged on the FTP server when the trace was created.
11. Press F11 (View 2) to view the date and time of the file you want to work with.
12. Verify that you are working with the most recent spooled file, QTMFFTRC.

Indicate in the problem report that the trace was tried and it failed. Send whatever trace information there is with the problem report.

The following example uses the FTP server DEBUG subcommand:

```

File Transfer Protocol

Previous FTP subcommands and messages:
Connecting to host name xxxxxnnn.xxxxxxxx.xxx.xxx at address
n.nnn.nn.nnn using port 21.
220-QTCP at xxxxxnnn.nnnnnnnn.nnn.nnn.
220 Connection will close if idle more than 5 minutes.
215 i5/OS is the remote operating system. The TCP/IPversion is
"V4R4M0".
>
331 Enter password.
230 TEST logged on.
250 Now using naming format "0".
257 "QGPL" is current library.

Enter an FTP subcommand.
===> quote debug

F3=Exit      F6=Print      F9=Retrieve
F17=Top      F18=Bottom    F21=CL command line

```

## Trace the FTP server with the Trace TCP/IP Application (TRCTCPAPP) command

The Trace TCP/IP Application (TRCTCPAPP) command allows *system-wide* tracing of *all* the FTP servers.

The TRCTCPAPP command is provided specifically for trained service and development personnel. \*SERVICE special authority is required to use this command. Use TRCTCPAPP in situations that require the capturing of trace data for service and development use. This command allows experienced personnel to dynamically start and stop tracing for applications.

With the use of TRCTCPAPP, trace information can be captured for the FTP TCP/IP application:

- Internal trace information can be captured for the iSeries FTP server. The information that can be captured for the FTP server can be filtered using remote IP address and port or iSeries user profile. Only one trace can be active at a time on the system.

Here are two examples of the use of the TRCTCPAPP command:

**Example 1:**

```
TRCTCPAPP APP(*FTP) SET(*ON)
```

This will start tracing for all FTP servers. Tracing for all other TCP applications is not affected.

**Example 2:**

```
TRCTCPAPP APP(*FTP) SET(*CHK)
```

This command is used to check the status of the tracing for the FTP server job(s). Assume that the last command entered was: >

```
TRCTCPAPP APP(*FTP) SET(*ON) USER(JOEC00L)
```

The format of the response to this command would be a set of messages that would look similar to the following commands:

```
TCP45B7 TRCTCPAPP APP(*FTP) SET(*ON) USER(JOEC00L)
      MAXSTG(*DFT) TRCFULL(*WRAP)
TCP45B1 Tracing active for *FTP.
TCP45B2 Data capture begun for *FTP.
TCP45B3 Data buffer wrapped for *FTP.
```

**Related concepts**

“Materials required for reporting FTP problems” on page 146

This topic describes the information that the IBM service representative might require to resolve an FTP problem.

**Related reference**

“DEBUG (Turn on the FTP Server Trace)” on page 44

## Trace the FTP client

You can use the DEBUG subcommand to track down problems on the FTP client.

To produce an FTP client trace or display the subcommands sent to the FTP server, use the DEBUG FTP client subcommand. The DEBUG subcommand toggles the debugging mode. If an optional debug-value is specified, it is used to set the debugging level. When debugging is on, each subcommand sent to the server is displayed and preceded by the string '>>>'. The debug-value must be set to 100 to produce an FTP client trace.

DEBUg [*debug value*]

*debug value*

If the debug-value is 0, debugging is off. If the debug-value is a positive integer, debugging is on.

If no value is specified, the debug-value is toggled from zero to one or from a positive integer to zero.

- 100** Initiate an FTP client trace. The client continues running the trace until DEBUG is turned off or until the FTP client is ended. (When the trace is ended, there may be a significant delay while the trace data is formatted.)

**Note:** The FTP client trace should only be used for reporting software problems to IBM. System performance can be adversely affected by this function.

A new capability has been added to the FTP client for debugging for V4R4. This function is similar to the DEBUG 100 described above. When the client is started, it first checks for the existence of a data area named QTMFTPD100.

You need to create the dataarea QTMFTPD100 in the QTEMP library using this command:

```
CRTDTAARA DTAARA(QTEMP/QTMFTPD100) TYPE(*LGL)AUT(*USE)
```

If the QTMFTPD100 dataarea exists, then it will set the debug value to 100 and start an FTP client trace. The purpose of this capability is to enable FTP client debug traces to be done in those situations when an FTP client trace *cannot* be started by issuing the DEBUG 100 subcommand.

#### **Related concepts**

“Materials required for reporting FTP problems” on page 146

This topic describes the information that the IBM service representative might require to resolve an FTP problem.

## **Work with FTP server jobs and job log**

You can research FTP errors by obtaining a spooled file of the FTP server job log.

A copy of the FTP server job log can be required to obtain additional information about errors that occur on the FTP server. The FTP server automatically writes a server job log to a spooled file when it ends with an error.

A server job log can be written to a spooled file without ending the server by issuing the following subcommand from an FTP client:

```
QUOTE RCMD DSPJOBLOG
```

To obtain a copy of error messages written to the server job log, this subcommand must be issued after the error has occurred. You can then inspect the job log using the WRKSPLF command.

This technique is recommended in those cases where the reply message returned to the client from the server only provides minimal information about an error occurring on the server machine. For example, this method is useful for obtaining details about I/O errors that occur on the server machine.

If the error prevents the FTP server job log from being obtained by the method described here, enter the following command to force a spooled job log to be created for each FTP session:

```
CHGJOB JOB(QUSRSYS/QTMFTPS) LOG(4 00 *SECLVL)
```

Then recreate the scenario which causes the error. To restore the original job log behavior after obtaining the required data, enter the following command:

```
CHGJOB JOB(QUSRSYS/QTMFTPS) LOG(4 00 *NOLOG)
```

To have a spooled job log produced at the end of each FTP session and each time an FTP server ends (with or without an error), use the Change Job Description (CHGJOB) command as follows:

```
CHGJOB JOB(QUSRSYS/QTMFTPS) LOG(4 00 *SECLVL)
```

To get a spooled job log only when a server ends, use the CHGJOB command as follows:

```
CHGJOB JOB(QUSRSYS/QTMFTPS) LOG(4 00 *NOLIST)
```

## **FTP server jobs and job names**

The FTP server jobs are started when the STRTCP command is run and the FTP AUTOSTART parameter is set to \*YES, or when the STRTCP SVR command is run with a SERVER parameter of \*FTP or \*ALL.

These jobs run in the QSYSWRK subsystem and their purpose is to monitor for incoming FTP users. The format for the names of these jobs is QTFTPnnnnn. The nnnnn is the job number of the FTP server job submitting to this server.

To work with FTP server jobs, enter the following CL command:

```
WRKACTJOB JOB(QTFTP*)
```

**Related reference**

WRKSPLF

---

## Code license and disclaimer information

IBM grants you a nonexclusive copyright license to use all programming code examples from which you can generate similar function tailored to your own specific needs.

| SUBJECT TO ANY STATUTORY WARRANTIES WHICH CANNOT BE EXCLUDED, IBM, ITS  
| PROGRAM DEVELOPERS AND SUPPLIERS MAKE NO WARRANTIES OR CONDITIONS EITHER  
| EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OR  
| CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND  
| NON-INFRINGEMENT, REGARDING THE PROGRAM OR TECHNICAL SUPPORT, IF ANY.

| UNDER NO CIRCUMSTANCES IS IBM, ITS PROGRAM DEVELOPERS OR SUPPLIERS LIABLE FOR  
| ANY OF THE FOLLOWING, EVEN IF INFORMED OF THEIR POSSIBILITY:

- | 1. LOSS OF, OR DAMAGE TO, DATA;
- | 2. DIRECT, SPECIAL, INCIDENTAL, OR INDIRECT DAMAGES, OR FOR ANY ECONOMIC  
| CONSEQUENTIAL DAMAGES; OR
- | 3. LOST PROFITS, BUSINESS, REVENUE, GOODWILL, OR ANTICIPATED SAVINGS.

| SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF DIRECT,  
| INCIDENTAL, OR CONSEQUENTIAL DAMAGES, SO SOME OR ALL OF THE ABOVE LIMITATIONS  
| OR EXCLUSIONS MAY NOT APPLY TO YOU.



---

## Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Software Interoperability Coordinator, Department YBWA  
3605 Highway 52 N  
Rochester, MN 55901  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

- | The licensed program described in this information and all licensed material available for it are provided
- | by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement,
- | IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. \_enter the year or years\_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

---

## Programming Interface Information

This FTP publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.



---

## Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

- | AS/400
- | i5/OS
- | IBM
- | IBM (logo)
- | iSeries
- | OS/2
- | OS/400
- | RISC System/6000
- | RS/6000
- | S/390

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

---

## Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

**Personal Use:** You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

**Commercial Use:** You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.







Printed in USA