

6

FTP, File Transfer, and More

You're probably already familiar with using a File Transfer Protocol (FTP) client for downloading files from the Internet. What you may not know is that you can do a lot more with FTP when you are connected to an OS/400 FTP server. That's because the OS/400 rendition of FTP allows you to call programs or execute commands on your 400 from an FTP client session. This ability can be great in situations that require a program to be run before or after transferring a file. We'll examine how to make the most of this functionality to help to integrate your 400 with PCs and LANs.

FTP Basics

As with other FTP servers, logging into OS/400's implementation of FTP requires a user to enter a user name and password. Any valid OS/400 user can log in via FTP. Before you can access your 400 via FTP, you must ensure that the FTP server is configured and running. Use the Change FTP Attribute (CHGFTPA) command to display and, if necessary, change your FTP server configuration. Figure 6.1 shows the screen displayed by typing CHGFTPA and pressing F4 to prompt.

```

Change FTP Attributes (CHGFTPA)

Type choices, press Enter.

Autostart servers . . . . . *YES_          *YES, *NO, *SAME
Number of initial servers . . . 3_              1-20, *SAME, *DFT
Inactivity timeout . . . . . 300_             0-2147483647, *SAME, *DFT
Coded character set identifier 00819_         1-65533, *SAME, *DFT
Server mapping tables:
  Outgoing EBCDIC/ASCII table . *CCSID_      Name, *SAME, *CCSID, *DFT
  Library . . . . .           Name, *LIBL, *CURLIB

  Incoming ASCII/EBCDIC table . *CCSID_      Name, *SAME, *CCSID, *DFT
  Library . . . . .           Name, *LIBL, *CURLIB
Initial name format . . . . . *LIB_       *LIB, *SAME, *PATH
Initial directory . . . . . *CURLIB_     *CURLIB, *SAME, *HOMEDIR
Initial list format . . . . . *DFT_       *DFT, *SAME, *UNIX
New file CCSID . . . . . *CALC_         1-65533, *SAME, *CALC...

Bottom
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

```

Figure 6.1: The CHGFTPA command is used to configure the OS/400 FTP server.

This screen is used to configure the FTP server. The first option defines whether the server runs automatically at IPL. The next parameter defines the number of instances of the FTP server that are initially started. The Inactivity Timer parameter defines the amount of time that an FTP client can remain logged in without activity before the server closes the session. The Coded Character Set Identifier defines the code page to be used by each FTP client. The next two parameters define the outgoing and incoming ASCII-to-EBCDIC translation tables. These can be used to customize character translations. The Initial Name Format parameter defines the file naming format that is used when a client initially logs in. The *LIB option causes file names to be displayed in the format LIBRARY.FILE. The *PATH format shows files in the form of their relative path. The Initial Directory parameter identifies which library will be displayed when a client logs in. *CURLIB defines that the current library for the user will be used. *HOMEDIR causes the user's home directory to be displayed. The Initial List Format parameter defines the type of file list to be displayed at a client session. *DFT uses the OS/400 default listing format. *UNIX displays the file list in a format similar to that used by the UNIX file system. Finally, the New File CCSID parameter defines the code page to be used when creating a new file from within an FTP session.

If the FTP server is not set to auto start or is not running, it can be started using the Start TCP/IP Server (STRTCPSVR) command:

```
STRTCPSVR SERVER(*FTP)
```

Now we're ready to access the OS/400 FTP server. Let's log in to ensure that everything is running properly. In Windows, display a DOS window and type FTP followed by the IP address of your 400 and press Enter. You will be prompted for a user name and password. Once you've entered a valid user name and password, the FTP prompt is displayed as shown in Figure 6.2. Type DIR and press Enter to display a list of files in the current directory.

```
C:\>ftp 198.162.0.1
Connected to 198.162.0.1
220-QTCP at S1234ABC.
220 Connection will close if idle more than 5 minutes.
User (90.0.0.3:(none)): QPGMR
331 Enter password.
Password:
230 QPGMR logged on.
ftp> dir
```

Figure 6.2: An example of logging into the OS/400's FTP server.

The FTP command **remotehelp** is used to display a listing of the commands supported by the FTP server itself. If you are unfamiliar with FTP, a list of common FTP commands and their usage is shown in Table 6.1.

Table 6.1: Some Frequently Used FTP Commands

Command(s)	Description
asc	Use ASCII transfer mode.
bin	Use binary transfer mode.
cd	Change the current directory on the server.
get, recv	Download a file from the server.
lcd	Change the local directory on your client.
mkdir	Create a new directory on the server.
put, send	Upload a file to the server.
quote	Send a remote command to the server.

FTP Scripting

Files can be easily sent to and from the 400 using the **recv** and **send** commands. It is important to remember that when you send or receive a physical file from the 400, it is going to be basically a text file when received on the PC. That means that any fields containing packed numeric data will not be viewable. The solution to that problem is to create a “pre-download” program to copy the data into a file without packed fields. This is where remote commands come into play. Using the **quote** FTP command along with the RCMD FTP server command, you can run a program on the 400 from within FTP as follows:

```
quote RCMD CALL PGM(XXX000CL)
```

In this example, the program named XXX000CL is called through the FTP session. This ability becomes even more powerful when you consider that you can create a “scripted” FTP session to run frequently used functions. One thing to consider is that the library list used by the FTP session may not meet the requirements of your program. Not to worry however, because we can use the same **quote** RCMD function to add any required libraries to the current job’s library list prior to running the application. The example shown in Figure 6.3 adds three libraries to the FTP job’s library list and then calls the required program.

```
quote RCMD ADDLIBLE QFNTCPL
quote RCMD ADDLIBLE QSRV
quote RCMD ADDLIBLE MYLIB
quote RCMD CALL PGM(BLD001CL)
recv BLD001PF
quit
```

Figure 6.3: A series of FTP commands used to run a program prior to downloading a file.

In this example, the Add Library List Entry (ADDLIBLE) command is used to add the libraries QFNTCPL, QSRV, and MYLIB to our library list. Next the CALL command is used to run the CL program BLD001CL. Then the FTP **recv** command is used to download the file BLD001PF from the system. Finally the **quit** command ends the FTP session. You can use practically any OS/400 command from within an FTP script. Besides simply calling a program, you can do things like send messages to a user or run a query. For troubleshooting purposes, you can use the following command:

```
ftp> Quote RCMD DSPJOBLOG
```

This causes the job log for the FTP session to be sent to the default printer. By creating *scripted FTP sessions*, you can create functions for commonly used tasks. FTP scripts are basically text files that contain all of the keystrokes to be sent to the FTP server. This includes the user name and password so it's usually a good idea to create a user profile specifically for FTP scripting purposes. This allows you to give that profile access to required objects only. When the script is run, all output that would normally be sent to the FTP client display is output a QSYSPRT printer file. This is also a great tool for troubleshooting problems within an FTP script

On the client side, you can use the “!” command to simply display a DOS prompt, or this command can be followed by a program name to launch that program. For example, the following command launches Windows Explorer and displays the C:\Windows directory:

```
ftp> ! EXPLORER C:\WINDOWS
```

If a DOS program is run, processing of the FTP script is halted until the DOS program completes. If a Windows program is launched, the FTP script continues processing. This is an important point to remember if continued processing of the FTP script needs the launched application to complete. This gives you the ability to perform the same functionality client-side that the RCMD command gives you server-side. In addition to the RCMD server-side FTP command, there are several other server-side commands supported by the OS/400 FTP server. A partial list of these is shown in Table 6.2. These include commands to create a library or physical file and commands to provide information about an object or objects on the server or about the server itself.

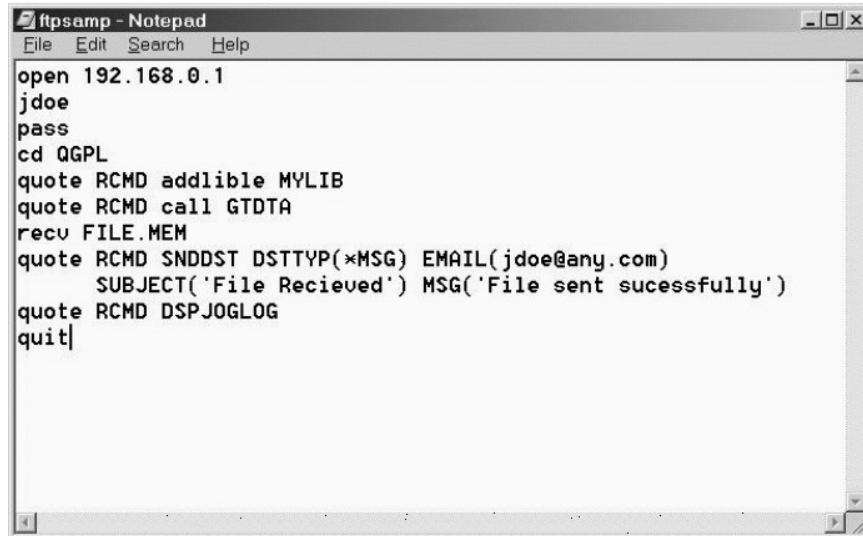
Table 6.2: OS/400 FTP Server Side Commands

Command(s)	Description
ABOR	End a data transfer in process.
ADDM FILE(library/file) MBR(member)	Add a member to a physical file.
ADDV FILE(library/file) MBR(member)	Add a variable length member to a physical file.
APPE (file name)	Append to a file if one exists.
CDUP	Move current directory to parent directory.
CRTL (library name)	Create a library.
CRTP FILE(Library/file) RCDLEN(80) MAXMBRS(*NOMAX)	Create a physical file.
CRTS FILE(Library/file)	Create a source physical file.
CWD	Change the default library.
DELE library/file.member	Delete a file member.
DLTF	Delete a file.
DLTL	Delete a library.
LIST (library/file.member)	Provide information about a specific object or objects in a specified library.

Table 6.2: OS/400 FTP Server Side Commands, *continued*

Command(s)	Description
MKD	Create a directory.
MODE (S=Stream, B=Block)	Specify data transfer mode.
NOOP	Performs no function. Simply sends 'OK' reply.
PWD	Return the name of the current library.
QUIT	End the FTP session.
RCMD (OS400 command)	Run a command on the server.
REIN	Reinitialize the current session.
RETR (file.member)	Retrieve data from the specified file/member.
RMD	Remove a directory.
RNFR from name	Use with RNTO to rename a file. Specifies the file to be renamed.
RNTO to name	Use with RNFR to rename a file. Specifies the new name for the file.
SITE (TRIM, LISTFMT, NULLSFLDS, CRTCCSID, NAMEFMT)	Use with specified parameters to set server- specific options.
STAT	Provide status information for the FTP session.
STOR	Store data to a file and replace the contents if the file exists.
STOU	Store data to a file and use a unique name if the file exists.
SYST	Return the name of the remote operating system.
TIME	Set the server timeout value for this FTP session.

Any of these server-side commands can be sent from an FTP script by using the **quote** command. Let's go through the process of creating a simple FTP script. Launch Windows Notepad by clicking Run from the start menu. Now type NOTEPAD.EXE and press Enter. When the notepad window is displayed, type the code shown in Figure 6.4.



```
ftpsamp - Notepad
File Edit Search Help
open 192.168.0.1
jdoe
pass
cd QGPL
quote RCMD addlib MYLIB
quote RCMD call GTDTA
recv FILE.MEM
quote RCMD SNDST DSTTYP(*MSG) EMAIL(jdoe@any.com)
SUBJECT('File Recieved') MSG('File sent sucessfully')
quote RCMD DSPJOBLOG
quit|
```

Figure 6.4: This example uses the RCMD FTP subcommand to execute OS/400 commands.

This example first changes the current library to QGPL, using the client-side **cd** command. Next we add MYLIB to the library list for our FTP session. We then call the program GTDTA and use the **recv** client command to download the specified file. Next we use the OS/400 Send Distribution (SNDST) command to send an e-mail message that the file was sent to a specified user. Finally we output the job log using the OS/400 Display Job Log (DSPJOBLOG) command and end the FTP session using the **quit** client command. Save this file as FTPSAMP.SCP and exit Notepad. Now display a DOS window and type the following command to launch an FTP session using our script file:

```
C:>FTP -s:ftpsamp.scp
```

The **-s** switch on the FTP command is used to identify that the FTP client should send the contents of the specified file rather than display an interactive FTP prompt.

FTP scripting can be extremely useful when trying to integrate a PC-based database, like SQL Server or Microsoft Access, with the 400. While the FTP script is

running commands on the 400, a program on the client is initiating it, thus you can have a PC program that exports data from a SQL Server database to an ASCII file. The program can then initiate an FTP script to send the file to the 400 and call a CL program to copy the data out of the flat ASCII file into a DB2/400 physical file. Another example would be a PC application that first runs an FTP script that calls a 400 program to create a specific set of data—daily sales for example—and then downloads the data to an ASCII file. The PC application then imports this data into a Microsoft Access database to be used for sales reporting.

Although it may seem that this could be accomplished using ODBC or OLE DB, this option is not always available. Also, when you consider the ability to run OS/400 commands, you can see how running FTP scripts from a PC client can be extremely useful.

The iSeries 400 as an FTP Client

In addition to having a robust FTP server, the iSeries also gives you an FTP client that can be used to perform similar functions using the 400 rather than a PC. The FTP client on the 400 is launched by typing `FTP` or `STRTCPFTP` from a command line, followed by the IP address of the FTP server enclosed in single quotes. Figure 6.5 shows the OS/400 FTP client display.

One nice thing that you'll notice is that the user name prompt from the FTP server will, if no user name is entered, send the current user name from the 400. Most of the same client commands supported by a PC client are supported on the 400. The **help** command can be used to display complete help text for the FTP client.

```
File Transfer Protocol

Previous FTP subcommands and messages:
Connecting to remote host 90.0.0.4 using port 21.
220 mail Microsoft FTP Service (Version 4.0).

Enter login ID (mfaust):
===> _____
_____
_____

F3=Exit      F6=Print      F9=Retrieve
F17=Top      F18=Bottom    F21=CL command line
```

Figure 6.5: The OS/400 FTP client login display.

OS/400 FTP scripting is done by first creating a member in a source physical file that contains all of the commands required for the FTP session. These commands are preceded by the user name and password to log into the FTP server. Figure 6.6 shows a sample FTP script that first logs into the FTP server using the user name ANONYMOUS, then sends the password a@b.c. (This satisfies the requirement, which is in place on most FTP servers, that an e-mail address be sent to the FTP server as the anonymous password.)

```
Anonymous
a@b.c
asc
send SALES.Jan
recv POS.txt POSDATA
quit
```

Figure 6.6: This is a sample of what an OS/400 FTP script looks like.

The next line uses the FTP command **asc** to ensure that transfers are in ASCII mode. On the following line, the **send** command is used to transfer the file SALES with the member name Jan to the FTP server. As you can see, the physical file member is referred to in the format file.member. On the next line we use the **recv** command to download the file POS.txt into the physical file POSDATA. By default, when downloading into an existing physical file, the member specified, or the first member if no member is specified, is replaced. The final command in this script is **quit**, which is used, as you would expect, to log out of the FTP server.

The actual process of running FTP from a batch job is accomplished by doing an Override Database File (OVRDBF) command for two files used by the **ftp** command. The first file named INPUT stores the commands to be sent to the FTP server. The second file is named OUTPUT and, upon completion of the FTP session, will contain the responses sent by the FTP server back to the session. Figure 6.7 shows a sample CL program that is used to run the FTP script shown earlier.

```

/*=====*/
/* To compile:                                     */
/*                                                */
/*          CRTCLPGM   PGM(XXX/FTP001CL) SRCFILE(XXX/QCLSRC) */
/*                                                */
/*=====*/
FTP001CL:   PGM

           OVRDBF   FILE(INPUT) TOFILE(MYLIB/FTPSCP) TOMBR(SENDDTA)
           OVRDBF   FILE(OUTPUT) TOFILE(MYLIB/FTPOUT) TOMBR(OUTPUT)

           FTP REMOTESYS(192.168.0.1)

           DLTOVR  OVR(*ALL)

           ENDPGM

```

Figure 6.7: This CL program is used to execute an FTP script from the 400.

In this example, we use the Override Database File (OVRDBF) CL command to specify the name of the file that will be used to send the client commands from

our FTP script. The second OVRDBF command overrides the output file to a specified file. This file can be very valuable as a debugging tool when the CL program running the FTP command has not achieved the expected results. Remember that scripts like this are somewhat dependent on the fact that each command sent completes successfully. More accurately, the scripts are oblivious to a command that does not complete successfully.

In addition to being able to do simple send/receive scripts like this one, you can also do more complex FTP scripts that do much more, as we saw earlier with PC clients. It's important to remember that the FTP server you are connecting to must support a function, running remote commands for example, for the function to be available on the client. Microsoft's FTP server does not support running remote commands. There are, however, third-party FTP server solutions that will give you this ability. Also remember that when you create a script on the 400, because the script is being run from a CL program, any processing that must occur as part of the process can simply be added to the CL program. This allows FTP scripts on the 400 to perform similar tasks to those written on a PC.

Error Handling with FTP Scripts

Error handling in a traditional sense does not exist within FTP scripting, because there is no ability to conditionally perform specific lines of an FTP script. There are, however, a few techniques that can be used to deal with errors if and when they do occur. The primary tool used for error handling is the OUTPUT file created when the FTP script is run. The downside to this method is that you can't deal with errors as they happen, but must deal with them after the fact by reading through the OUTPUT file and specifically scanning for returned errors by the reply code.

An example of how this is done in RPG is shown in Figure 6.8. The sample program reads through each record in the OUTPUT file and uses the %SCAN function to locate the error code 550, which is returned if a file to be sent or received does not exist. This application then prints out an error log for the job, but your application might handle this in some other way, possibly running an application to build some data, or sending a message to the QSYSOPR message queue.

```

=====
* To compile:
*       CRTBNDRPG  PGM(XXX/SLG001RG) SRCFILE(XXX/QRPGLESRC)
*
=====
* FTP Session Output file
FOUTPUT  IF  E          DISK
FQPRINTS 0  F  80      PRINTER OFLIND(*INOF)
* Read Through Output file until end of file
C          Eval        *INOF=*ON
C          DoU         %EOF(OUTPUT)
C          Read        OUTPUT
C          If          %EOF(OUTPUT)=*OFF
C          If          %Subst(SRCDTA:1:4)='550 '
C          If          *INOF=*On
C          Except     Header
C          EndIf
C          Except     Error
C          EndIf
C          EndIf
C          EndDo
C          Eval        *INLR=*ON
C          Return
OQPRINTS E          Header          2  2
O          30 'FTP Session Error Report'
O          E          Error          1
O          SRCDTA          80

```

Figure 6.8: The RPG program SLG001RG searches an FTP OUTPUT file for a specific error.

Using a slightly modified version of our earlier example, we call this program from the CL program used to initiate the FTP script. Figure 6.9 shows a sample of what this program looks like. Because the override for the OUTPUT file is already in place, all we really need to do here is call our report program.

The important thing to remember is that, as with this example, all handling of errors is done after the fact. This can affect the way you design an application that uses FTP scripting. That point aside, this is still a very powerful tool. Being able to script FTP sessions from an iSeries batch job allows you to perform data transfers that are initiated from the iSeries, rather than having to be run on a PC.

```
/*=====*/  
/* To compile: */  
/* */  
/* CRTCLPGM PGM(XXX/FTP002CL) SRCFILE(XXX/QCLSRC) */  
/* */  
/*=====*/  
FTP002CL: PGM  
  
OVRDBF FILE(INPUT) TOFILE(MYLIB/FTPSCP) TOMBR(SENDDTA)  
OVRDBF FILE(OUTPUT) TOFILE(MYLIB/FTPOUT) TOMBR(OUTPUT)  
  
FTP REMOTESYS(192.168.0.1)  
  
CALL SLG001RG  
  
DLTOVR OVR(*ALL)  
  
ENDPGM
```

Figure 6.9: This CL program runs an FTP script and then prints out any 550 errors received.

How you determine whether your project is best suited to an FTP job that runs on the 400 or on a PC depends greatly on what is required. For example, if your FTP project requires programs to be run both on the 400 and on the PC, you'll probably want to initiate it from the PC's FTP client. If the FTP job is to be part of an existing job stream on the 400, then it makes more sense to keep the FTP script running from the OS/400 FTP client. Whether you run the FTP script from a PC client or from OS/400's FTP client, FTP can do a lot more than just file transfers.