FTP4W API User Manual

License

Thanks To Santanu Lahiri for providing the source of WinFTP, a FTP client for windows. I have learned a lot (about FTP and Windows) by reading it.

Thanks to all the Internet community which has reported some implementation problems and has contributed to this new release, especially:

- Gillian Duncan (Corrections of this manual)
- Burks Oakley <b-oakley@uiuc.edu> (tests and ToolBook support)
- Richard Terpstra <terpstr2@ksla.nl> and Kees de Rooij (VB Sample)
- Vince Vielhaber <vev@msen.com> (VMS and C++ support)
- Bram Buitendijk <Bram.Buitendijk@library.KNAW.nl> (MS-Access support)
- Teemu Mottonen <Teemu.Mottonen@ktl.fi> (Corrections of this manual)
- Robin Bowes <robin@plato.ucsalf.ac.uk> (Paradox supports)
- Terry Field <terry@mincom.oz.au> (Bug report)
- David Combs <dkcombs@netcom.com> (has fully rewritten this manual)
- Andreas Tikart <Andreas.Tikart@uni-konstanz.de> (tp7 sample)
- Ricky Freyre <rickyf@mail.halcyon.com>

Overview

FTP4W.DLL provides an implementation of the FTP protocol (specified in the RFC 959).
It is a Windows Dynamic Library (DLL), which can be used by any language (and any compiler). It requires a Windows Sockets DLL (Winsock.DLL).

FTP4W provides four groups of functions:

Local Functions
Connection functions
Data transfer functions
FTP Commands functions

The data transfer functions can be used in two modes:

If the application chooses the synchronous mode (set by the FtpSetSynchronousMode function), all the FTP4W calls will return when the task is finished. Each function returns an integer return-code.

If the application chooses the asynchronous mode (set by FtsetAsynchronousMode), the data transfer function (and FtpLogin) will return before the task has been done. The application must wait for a message posted by the DLL when the job is over.
The message contains two arguments wParam and lParam (please refer to a Windows programmer's reference) which are used to pass information such as return codes.
The functions return an integer which is FTPERR_OK if the request is accepted, and an error code such as FTPERR_NOTINITIALIZED if it is rejected (in this case the application will receive no message).

Synchronous functions have been implemented because some languages can not handle user defined messages, but it is recommended to use asynchronous versions.

Asynchronous calls give the application a way to follow the progress of a data transfer.
The DLL posts a message for the application each time it receives a packet of data. This message contains two arguments:

wParam: FALSE (operation not completed)
lParam : number of bytes received/sent

The FTP4W calls do not need any handle to identify the FTP session. Rather, they use the Windows function **GetCurrentTask** to get a task identifier. This mechanisms avoids the use of a argument but it prohibits having more than one FTP session for a given task (note that if the same application is started twice, FTP4W will just see two differents tasks, so each application can have its own FTP session).

Known bugs

This version of Ftp4w does not support the following stacks:

- SPRY winsockets
- LAN Workplace (Novell)

To run Ftp4w with LAN Workplace, the file Ftp4w.Dll should be replaced by Ftp4wLWP.Dll.

Important changes

Since version 2.3, the WEP function does not free any resources.

Programming with the FTP4W API

To use the FTP4W functions following files are provided:

- This reference
- The Help file written by Michael Douglass
- The DLL FTP4W.DLL
- The DLL to be used with Lan Workplace FTP4WLWP.DLL
- The 32 bits version FTP4W32.DLL
- A C-header file Ftp4w.h
- A Visual Basic header file Ftp4w.Vb
- A Toolbox header file Ftp4w.Tbx
- A library file FTP4W.LIB

The first function that an application should call is **FtpInit**. It allocates buffers and get some information about the task which has called it.

If the application wishes (or must) use the synchronous mode, it must *now* call the function **FtpSetSynchronousMode**.

The task is ready to make a connection with a FTP server. It must either
- Call **FtpOpenConnection**, **FtpSendUserName** and **FtpSendPasswd**
- or just call **FtpLogin** (which combines the 3 functions).

If it succeeds the user is logged on and can use any of the other FTP4W functions. For Instance, the application can call **FtpDir** to read the contents of the remote directory.

To end the connection, the application must call **FtpCloseConnection**. If the function does **not** succeed (e.g. the network has been shut down), it must call **FtpLocalClose**.

In both cases, to release the allocated buffers, the application must call **FtpRelease** before it exits.

The FTP4W functions

This table lists alphabetically all the functions implemented in this version. The remainder of this chapter describes them one by one.

Ftp4wVer	Gives the 2-part version of the DLL (packed into an int).
FtpAbort	Aborts the current data transfer
FtpAppendToLocalFile	Appends a remote file onto a local file
FtpAppendToRemoteFile	Appends a local file onto a remote file
FtpBytesToBeTransferred	Gives the length of the file which is to be received
FtpBytesTransferred	Gives number of bytes which have been received
FtpCDUP	"CD's" remote default dir UP to its parent directory
FtpCloseConnection	Ends a FTP session
FtpCWD	Changes the remote default directory
FtpDataPtr	Gives a pointer to the internal Ftp4w structure
FtpDeleteFile	Deletes a remote file
FtpDir	Gets the remote directory
FtpGetFileSize	Obsolete: see instead FtpBytesToBeTransferred
FtpHelp	Gets the help file of the host's FTP server
FtpInit	First function to be called
FtpIsAsynchronousMode	Checks if Ftp4w is in asynchronous mode
FtpLocalClose	Closes local sockets
FtpLogin	Combines Ftp-OpenConnection,SendUserName,SendPasswd
FtpLogTo	Enables/Disables logs
FtpMKD	Creates a remote directory
FtpOpenConnection	Makes an FTP connection
FtpPWD	Gets the remote default directory
FtpQuote	Sends a user-defined command to the server
FtpRecvFile	Retrieves a remote file
FtpRelease	Last function to be called, frees local resources
FtpRestart	Checks if the RESTART command is implemented
FtpRestartRecvFile	Receives an opened file from a given position
FtpRestartSendFile	Sends an opened file from a given position
FtpRMD	Removes a remote directory
FtpSendAccount	Sends user's account
FtpSendFile	Sends a local file to the remote host
FtpSendPasswd	Sends user's password
FtpSendUserName	Sends username
FtpSetAsynchronousMode	Switches to Asynchronous mode
FtpSetDefaultPort	Changes default FTP port
FtpSetDefaultTimeOut	Changes default time out
FtpSetNewDelay	Changes the delay between N frames
FtpSetNewSlices	Changes the above "N frames" number
FtpSetPassiveMode	Set passive or active mode
FtpSetSynchronousMode	Switches to synchronous mode (default)
FtpSetType	Changes the data representation type
FtpSetVerboseMode	Set verbose or silent mode
FtpSyst	Asks for the host system

Ftp4wVer

Ftp4w returns the version number of the DLL, as an integer. The low order byte is the release number, the high order byte is the major version number.

The function copies in the users's buffer a string which contains information on the DLL (name, version, author, copyright). This string is guaranteed not to exceed 100 characters.

Syntax: Ftp4wVer(LPSTR szVerStr, int nStrSize)
(LPSTR is a 32-bits pointer)

Arguments:	szVerStr:	a buffer which is to receive the version information
	nStrSize:	its size

Return: an integer which contains the version.

FtpAppendToLocalFile / FtpAppendToRemoteFile

See FtpRecvFile or FtpSendFile.

FtpAbort

This function aborts a data transfer without breaking the connection.

This function returns immediately. The data transfer is actually aborted somewhat later, at which time that prior data transfer function (FtpDir, FtpRecvFile, FtpSendFile) returns.

It will return (either by return or PostMessage) a special error code (FTPERR_CANCELBYUSER) which means that the transfer has been aborted.

The opened files are closed but not removed.

Syntax: FtpAbort ()

Return Codes:

FTPERR_OK	Abort is in progress

FtpBytesTransferred / FtpBytesToBeTransferred

Syntax:
long FtpBytesTransferred (void)
long FtpBytesToBeTransferred (void)

FtpBytesTransferred returns the number of bytes which has been transferred. This number is reset for each new transfer; i.e. it is not cumulative.

FtpBytesToBeTransfered returns the total length of the file which is transfered. For ASCII transfers, it can be slightly different from the number of bytes to be received. Furthermore, if the result of this function is 0, it means that the FTP server did not send this information (Windows-NT server for instance).

Note: Since the previous versions spelled *Transfered* instead of *Transferred*, the functions *FtpBytesTransfered* and *FtpBytesTransfered* are still implemented for backwards compatibilty.

FtpCDUP

This function changes the remote default directory up to its parent directory.

Syntax: FtpCDUP ()

Return Codes:

FTPERR_OK	Directory has been changed
FTPERR_SERVERCANTEXECUTE	CWD has failed (directory does not exists..)
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpCloseConnection

This function tries to close gracefully the connection. It will not succeed if a file transfer is in progress or if the server has timed-out. In this case, you must use FtpLocalClose.

Syntax: FtpCloseConnection (void)

Return Codes

FTPERR_OK	FTP session has been closed
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.

FtpCWD

This function changes the default directory on the remote server.

Syntax: FtpCWD (LPSTR szPath)

Argument: szPath: name of the new directory

Return Codes:

FTPERR_OK	Directory has been changed
FTPERR_SERVERCANTEXECUTE	CWD has failed (directory does not exists..)
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpDataPtr

FtpDataPtr returns the address of the internal structure LPProcData (see Ftp4w.h).
This function is provided only for debugging purposes and should be use cautiously.

FtpDeleteFile

This function deletes a remote file.

Syntax: FtpDeleteFile (LPSTR szFile)

Argument: szFile: name of the file to be deleted

Return Codes:

FTPERR_OK	File has been deleted
FTPERR_FILELOCKED	File can not be deleted
FTPERR_NOREMOTEFILE	File has not been found
FTPERR_SERVERCANTEXECUTE	File can not be deleted
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpDir

This function reads the remote directory.

<u>* Asynchronous Mode</u>

It can be used in either of two ways:
- The function posts a message to the application each time a file name is received.
- The function fills a file with the file names and posts a message once the directory is terminated.

In the first case, the function posts a message with wParam=FALSE each time a data line has been received. lParam is a pointer on this line. The application must save the data because the next line sent by the server will overwrite it. The string is null-terminated and contains only one line (the ending <CR><LF> has been removed). The final message received by the application will have wParam=TRUE and lParam is the return code.

In the second case the dir is written in the file szFile. Once it is finished, FTP4W posts a message to the application.

<u>* Synchronous Mode</u>

The application must specify a file name, which will be filled with the remote directory. The functions returns an error code or FTPERR_OK if it is successful.
The two last arguments are ignored.

Syntax:
FtpDir (LPSTR szFilter, LPSTR szFile, BOOL bLongDir, HWND hWnd, WMSG wMsg);

Arguments:	szFilter	Remote path and filename mask. Note that the wildcard expansion is dependent of the remote host and is not necessarily the same as MS DOS format. An empty string or NULL will give the current remote directory.
	szFile	The file where the data is to be written, if szFile is NULL, the first mode is used (a message is posted each time a complete line has been received)
	bLongDir	Allow the application to choose between the long or the short form of listing. The short form give only the name of the files, the format of the long form depends on the server.
	hWnd	the handler of the windows to which the message is to be passed
	wMsg	the application-defined message to be passed to the application

Return Codes:

FTPERR_OK	Dir has been done
FTPERR_PASVCMDNOTIMPL	Server does not support passive mode
FTPERR_NOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server rejects the command TYPE ASCII
FTPERR_CANTOPENFILE	Local file can not been open
FTPERR_CANTWRITE	FTP4W can not write in local file (disk full)
FTPERR_CANTCREATESOCKET	No more free sockets (Two sockets are needed)

FTPERR_TRANSFERREFUSED	the server refused the dir command
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpGetFileSize

Warning: This function has been obsoleted by FtpBytesToBeTransferred.

This function tries to get the size of the file which is to be received. It must be used immediatly after a FtpRecvFile, because it searches in the most recent reply, looking to see if the server has sent back the size of the file.

If the function succeeds, it returns the length of the file. (Note that in ASCII mode, it can be slightly different from the number of bytes FTP4W will receive). Otherwise, it returns 0.

This function is obsolete and should be replaced by FtpBytesToBeTransferred.

Syntax
FtpGetFileSize()

returns DWORD.

FtpInit

FtpInit must be called before any other function. It allocates buffers, reads information about the task which has called it and creates an invisible window for its internal use.
Before it exits, the application must call **FtpRelease** to release these internal resources.

The function requires the handler of an application window (or NULL).

Syntax: FtpInit (HWND hParentWnd)

Argument: hParentWnd is the handle of an existing application window.

Return codes:

FTPERR_OK	Initialisation has been done
FTPERR_INSMEMORY	not enough memory
FTPERR_CANTCREATEWINDOW	FtpInit can't create its window
FTPERR_SESSIONUSED	The task has already a FTP4W session

FtpIsAsynchronousMode

This function checks if Ftp4w is in asynchronous mode.

Syntax:	FtpIsAsynchronousMode()
Return:	TRUE if Ftp4w is in asynchronous mode, FALSE if Ftp4w is in synchronous mode.

FtpLocalClose

This function closes the opened socket without warning the server. You must use this function only if FtpCloseConnection has failed.

Syntax:	FtpLocalClose (void)
Return Codes:	FALSE if the session has not been initialized by FtpInit Otherwise TRUE.

FtpLogin

This function combines the three preceding functions. It completes the login procedure.
If the current mode is "synchronous", FtpLogin will return when the job is over, and the two last arguments are unused. Otherwise ("asynchronous mode") it immediatly returns FTPERR_OK, and then later, when the request is completed, the application will receive a wMsg message in the hWnd window.

The message will be followed by:
wParam: TRUE
lParam: The return code of the function

Syntax:
FtpLogin(LPSTR szHost,LPSTR szUser,LPSTR szPass, HWND hWnd,WMSG wMsg)

Arguments: szHost: name of the remote host (the computer on which the server is running)
szUser: name of the user
szPass: Password (it can be NULL if the user has no password)
hWnd is the handler of the windows to which the message is to be posted
wMsg is the application-defined message to be posted to the application

Return Codes:
Return codes are in the Low Word of the lParam argument:

FTPERR_OK	User is logged on
FTPERR_ENTERACCOUNT	Successful function but server awaits an account name
FTPERR_LOGINREFUSED	The USER/PASSWD has been rejected
FTPERR_NOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.
FTPERR_CANTCREATESOCKET	The socket has not been created
FTPERR_CONNECTREJECTED	Connect has been rejected (server is not a FTP server, ...)
FTPERR_CANTCONNECT	The connect has failed
FTPERR_TIMEOUT	The connect has timed-out

FtpLogTo

This function set or reset a log mode. In log mode, all data sent or received on the control port (21) are sent to the opened file passed as an argument.
The frame which contains the password is logged as "PASS +++" for obvious reasons.

To set silent mode (default), just call **FtpLogTo (HFILE_ERROR)**.
Note that the file is not closed by Ftp4w.

Syntax: FtpLogTo (HFILE hLogFile)

Argument:	hLogFile	A opened file handler to be written to. HFILE_ERROR to set silent mode.

FtpMKD

This function creates a directory on the remote server. The full name of the new directory is returned in a user's buffer. If the FTP-server has succesfully created the directory but did not return its full name, this buffer is set to an empty string.

Syntax: FtpMKD (LPSTR szPath, LPSTR szBuf, UINT uBufSize)

Argument:	szPath:	name of the directory to be created
	szBuf:	Buffer to be filled with the full name of the created directory
	uBufSize:	Size of the user's buffer

Return Codes:

FTPERR_OK	Directory has been created
FTPERR_SERVERCANTEXECUTE	MKD has failed (can not create directory, directory already exists, ...)
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpOpenConnection

This function establishes the connection with the FTP server.
Once the connection is done, it waits for the reply of the server.

This reply must begin with "220" (RFC 959), if not a special error is generated.

FTP4W does not check to see if a connection already exists.

Syntax: FtpOpenConnection (LPSTR szHost)

Argument: szHost: The name of the remote host to connect to

Return codes:

FTPERR_OK	Successful connection
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_CANTCREATESOCKET	The socket has not been created
FTPERR_CONNECTREJECTED	Connection has been rejected (server is not a FTP server, ...)
FTPERR_CANTCONNECT	The connection has failed
FTPERR_TIMEOUT	The connection has timed-out
FTPERR_NOREPLY	The connection is successful, but FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	The connection is successful and FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.

FtpPWD

This function returns the name of the default directory on the remote server.

Syntax: FtpPWD (LPSTR szPath, UINT uBufSize)

Argument:	szPath:	buffer to be filled with the name of the remote default directory
	uBufSize:	size of this buffer

Return Codes:

FTPERR_OK	Name of the remote directory available in the buffer
FTPERR_SERVERCANTEXECUTE	PWD has failed (directory does not exists..)
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_PWDBADFMT	FTP4W can not interpret server's answer
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpQuote

It allows the user to send to the server any FTP command he wants. FTP4W will send it to the server and waits for its reply.

Note: The names of the commands given there are DIFFERENT from the commands you would type in by hand with a text-oriented FTP client. In fact, the ones you type in by hand are read by a higher-level "front end" to the REAL ftp, and that higher-level then translates them into shorter names.
To have the list of the codes accepted by the FTP servers, refers to the RFC 959.

Note: The application can not start a data-transfer with this command.

The return code is either a FTP code (ie 200) or a FTP4W error code (FTPERR_NOREPLY, FTPERR_SENDREFUSED, ...).
If szReplyBuf is not NULL, The reply (if any) is copied into a user's buffer.

Syntax FtpQuote (LPSTR szCmd, LPSTR szReplyBuf, UINT uBufSize);

Arguments:	szCmd	The command to be sent
	szReplyBuf	The buffer to copy the answer
	uBufSize	The size of the user's buffer

Return Codes:	A Ftp4w's error code or a 3 digits number between 100 and 699.

FtpRecvFile / FtpAppendToLocaFile

Note : Since FtpAppendToLocalFile uses exactly the same syntax, only FtpRecvFile will be described. The only difference between them occurs when the local file already exists, in which case the Append function appends the remote file onto it, whereas the Recv function simply over-writes it.

This function copies a remote file to a local file. In the asynchronous mode, the function returns immediatly, then the application will receive a message when the transfer is completed with wParam=TRUE (transfer completed), lParam=return code. In the synchronous mode, the function returns when the transfer is completed.

In the notification mode, the application will receive a message each time some data has been received. The same message as above is used but wParam will be FALSE, lParam will be the current position in the file (it is also the number of bytes which have been received).

In synchronous mode, if bNotify has not been set, the final arguments hParentWnd and wMsg are not used.

If the local file already exists, The function FtpRecvFile over-writes it, whereas FtpAppendToLocalFile appends the remote file at the end of the file.

If the file does not exists, it is created in any case.

Syntax:

```
FtpRecvFile (LPSTR szRemote, LPSTR szLocal,
             char cType, BOOL bNotify,
             HWND hParentWnd, UINT wMsg)
FtpAppendToLocalFile (LPSTR szRemote, LPSTR szLocal,
                        char cType, BOOL bNotify,
                      HWND hParentWnd, UINT wMsg)
```

Arguments:

	szRemote	Remote file specification
	szLocal	The file where to write the data.
	cType	TYPE_A for ASCII, TYPE_B for binary
	bNotify	A message should be sent by to the application each time
	a frame	
		has been received.
	hWnd	the handler of the windows to which the message is to be
	passed	
	wMsg	the application-defined message to be passed to the
	application	

Return Codes:

FTPERR_OK	File received
FTPERR_PASVCMDNOTIMPL	Server does not support passive mode
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server has rejected the command TYPE
FTPERR_CANTOPENFILE	Local file can not been open
FTPERR_CANTWRITE	FTP4W can not write in local file (disk full)
FTPERR_CANTCREATESOCKET	No more free sockets (Two sockets are needed)
FTPERR_TRANSFERREFUSED	the server refused the Retrieve command
FTPERR_NOREPLY	FTP4W has received no reply.

	FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpRelease

FtpRelease must be called before the application exits. It frees all resources taken by FtpInit. The function requires no arguments.

Syntax: FtpRelease ()

return codes:

FTPERR_OK	Resources have been released
FTPERR_STILLCONNECTED	The connection is still active. Nothing has been done.

FtpRestart

This command makes the server to begin the next file transfer at the specified position. This command should be issued just prior a file transfer request, which is not possible with the high-level transfer functions. Therefore this is mostly an internal command.
It has been exported since some servers does not support this command. Thus it is an easy way to check the server before starting any file transfer.

Syntax: FtpRestart (long lByteTransfer)

Argument :	lByteTransfer	Position of the next file transfer. If this value is negative or 0, the function does nothing and returns FTPERR_RESTARTOK

return codes:

FTPERR_RESTARTOK	The command successful but it has no effect.
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection
FTPERR_CMDNOTIMPLEMENTED	Command not implemented

FtpRestartRecvFile

This command starts a file transfer from a specified position. Please refer to the FtpRecvFile command to have more info concerning file transfers.
This command should be used only in binary mode, since the position in a text file has little meaning.

```
Syntax:  FtpRestartRecvFile (LPSTR szRemote, HFILE hLocal, char cType,
                             BOOL bNotify, long lByteCount,
                             HWND hParentWnd, UINT wMsg);
```

Arguments :	szRemote	The remote file to be received	
	hLocal	A Windows handler to an opened file which is to be written	
		cType	TYPE_A for ASCII, TYPE_B for binary
		bNotify	A message should be sent by to the application each time
		a frame	
			has been received.
		lByteCount	Starting position of the transfer
		hWnd	the handler of the windows to which to pass the message
		wMsg	the application-defined message to pass to the application

```
Syntax:  FtpRestartSendFile (HFILE hLocal, LPSTR szRemote, char cType,
                             BOOL bNotify, long lByteCount,
                             HWND hParentWnd, UINT wMsg);
```

Return Codes:

FTPERR_OK	File received
FTPERR_PASVCMDNOTIMPL	Server does not support passive mode
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server has rejected the command TYPE
FTPERR_CANTOPENFILE	Local file can not been open
FTPERR_CANTWRITE	FTP4W can not write in local file (disk full)
FTPERR_CANTCREATESOCKET	No more free sockets (Two sockets are needed)
FTPERR_TRANSFERREFUSED	the server refused the Retrieve command
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.

FtpRestartSendFile

This command starts a file transfer from a specified position. Please refer to the FtpSendFile command to have more info concerning file transfers.
This command should be used only in binary mode, since the position in a text file has little meaning.

```
Syntax:  FtpRestartSendFile (HFILE hLocal, LPSTR szRemote, char cType,
                             BOOL bNotify, long lByteCount,
                             HWND hParentWnd, UINT wMsg);
```

Arguments :	hLocal	A Windows handler to an opened file which is to be read. Ftp4w starts reading this file from the current position.	
	szRemote	The remote file to be written from the position lByteCount.	
		cType	TYPE_A for ASCII, TYPE_B for binary
		bNotify	A message should be sent by to the application each time a frame has been received.
		lByteCount	Starting position of the transfer
		hWnd	the handler of the windows to which to pass the message
		wMsg	the application-defined message to pass to the application

```
Syntax:  FtpRestartSendFile (HFILE hLocal, LPSTR szRemote, char cType,
                             BOOL bNotify, long lByteCount,
                             HWND hParentWnd, UINT wMsg);
```

Return Codes:

FTPERR_OK	File has been sent
FTPERR_PASVCMDNOTIMPL	Server does not support passive mode
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server rejects the command TYPE ASCII
FTPERR_CANTOPENFILE	Local file can not been open
FTPERR_CANTWRITE	FTP4W can not write in local file (disk full)
FTPERR_CANTCREATESOCKET	No more free sockets (Two sockets are needed)
FTPERR_TRANSFERREFUSED	the server refused the STOR command
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpRMD

This function removes a directory from the remote server.

Syntax: FtpRMD (LPSTR szPath)

Argument: szPath: name of the directory to be deleted

Return Codes:

FTPERR_OK	Directory has been removed
FTPERR_SERVERCANTEXECUTE	RMD has failed (directory is not empty)
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpSendAccount

This function sends the account to the server. This function should be used when FtpLogin or FtpSendPasswd return FTP_ENTERACCOUNT.

Syntax: FtpSendAccount (LPSTR szAccount)

Argument: szAccount: Account information

Return Codes:

FTPERR_OK	User is logged on
FTPERR_LOGINREFUSED	The USER/PASSWD/ACCOUNT has been rejected
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_NOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.

FtpSendFile / FtpAppendToRemoteFile

Note : Since FtpAppendToRemoteFile uses exactly the same syntax, only FtpSendFile will be described. The only difference between them occurs when the remote already exists, in which case the Append function appends the local file onto it, whereas the Send function simply over-writes it.

This function copies a local file to a remote file.
The application will receive a message when the transfer is completed with wParam=TRUE (transfer completed), lParam=return code. In the synchronous mode, the function returns only after the transfer has been completed.

In the notification mode, the application will receive a message each time some data has been sent. The same message as above is used but wParam will be FALSE, lParam will be the current position in the file (it is also the number of bytes which have been sent).

In synchronous mode, if bNotify has not been set, the final arguments (hParentWnd and wMsg) are not used.

Syntax:

```
FtpSendFile (LPSTR szLocal, LPSTR szRemote,
             char cType, BOOL bNotify,
             HWND hParentWnd, UINT wMsg)
FtpAppendToRemoteFile (LPSTR szLocal, LPSTR szRemote,
                       char cType, BOOL bNotify,
                       HWND hParentWnd, UINT wMsg)
```

Arguments:

szLocal	The file to be sent
szRemote	Remote file specification
cType	TYPE_A for ASCII, TYPE_B for binary
bNotify	A message should be sent by to the application each time a frame has been received.
hWnd	the handler of the windows to which to pass the message
wMsg	the application-defined message to pass to the application

Return Codes:

FTPERR_OK	File has been sent
FTPERR_PASVCMDNOTIMPL	Server does not support passive mode
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server rejects the command TYPE ASCII
FTPERR_CANTOPENFILE	Local file can not been open
FTPERR_CANTWRITE	FTP4W can not write in local file (disk full)
FTPERR_CANTCREATESOCKET	No more free sockets (Two sockets are needed)
FTPERR_TRANSFERREFUSED	the server refused the STOR command
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpSendPasswd

This function sends the password to the server.

Syntax: FtpSendPasswd (LPSTR szPasswd)

Argument: szPasswd: Password of the user

Return Codes:

FTPERR_OK	User is logged on
FTPERR_ENTERACCOUNT	Successful function but server awaits an account name.
FTPERR_LOGINREFUSED	The USER/PASSWD has been rejected
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_NOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.

FtpSendUserName

This function sends the user's name to the server. This authentification is necessary to begin a file transfer.
This function will usually return the "error-like" return value: FTPERR_ENTERPASSWORD (Successful function but server awaits a password). It means only that everything is just fine, it is simply reminding them that it still needs the password.>

Syntax: FtpSendUserName (LPSTR szUserName)

Argument: szUserName: Name of the user

Return Codes:

FTPERR_OK	User is logged on
FTPERR_ENTERPASSWORD	Successful function but server awaits a password.
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection.

FtpSetDefaultPort / FtpSetDefaultTimeOut

Syntax:
int FtpSetDefaultPort (int nDefPort)
int FtpSetDefaultTimeOut (int nTimeOutInSeconds)

These functions are used to change either the FTP-control port (21 by default) or the timeout (30 seconds by default).
Note that the FTP-data potr can not be changed.

The new timeout is given in seconds.

FtpSetNewDelay / FtpSetNewSlices

Syntax:
```
int  FtpSetNewDelay  (int nNewDelayInMilliseconds)
int  FtpSetNewSlices (int nSliceAlone, int nSliceMultiUser)
```

When a given number of frames has been received during a data transfer, FTP4W will wait for a while in order to let other tasks run.

FtpSetNewDelay allows the application to change the length of the pause. The argument is the length of the pause to be applied in milliseconds.

FtpSetNewSlices is used to change the number of frames which will cause a pause. The first argument is the number of frames when one FTP session are active, the second argument is used when two or more sessions are active.
Both arguments should not be set to zero.

FtpSetPassiveMode

Syntax:
void FtpSetPassiveMode (BOOL bPassif)

These command requests the FTP-server to "listen" on a data port and to wait for a connection rather than initiate one upon receipt of a transfer command.

These command is not implemented on all FTP-server, and thus the application must check the return code of the next data-transfer (FtpRecvFile, FtpSendFile, FtpDir).

Note: If FtpInit has not been called, these calls will cause a GPF.

Arguments:	bPassif	TRUE if the application wants to switch to passive mode, FALSE if it wants to reset the default mode.

Return codes:

FTPERR_OK	Mode has been changed

FtpSetType

This function changes the default transfer type.

Syntax: FtpSetType (char cType)

Argument: cType: new default transfer mode (either TYPE_A or TYPE_I)

Return Codes:

FTPERR_OK	Type has been changed
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

FtpSetVerboseMode

If a programmer needs to have a look on each frame sent by the server, he uses this function. He will get a message (by the **SendMessage** function) each time a frame has been received. The argument wParam is TRUE, lParam points to the frame. It is NULL-terminated but can contain p more than one line (a line is ended with <CR><LF>).

Note that the frame will be overwritten by the next reply from the server.

Syntax: FtpSetVerboseMode (BOOL bVerboseMode, WND hWnd, WMSG wMsg)

Arguments:

bVerboseMode	TRUE if the application wants to watch incoming messages, FALSE to end a previous FtpSetVerboseMode
hWnd	the handler of the window to which the message is to be passed
wMsg	the application-defined message to be passed to the application each time a frame has been received.

Return codes

FTPERR_OK	Mode has been changed
FTPERR_NOTINITIALIZED	session has not been initialized by FtpInit

FtpSyst

This command asks to the server the system on which it is running.

The return code is either a FTP4W error code (FTPERR_NOREPLY, FTPERR_SENDREFUSED, ...). or the index into the array of strings passed as an argument.

The argument given is an array of pointers to string. Each string should contain a possible system name. The final pointer must be NULL The function returns the index of the string whose contents matches the answer returned by the server. If no system name has been found, FTPERR_SYSTUNKNOWN is returned.

In the given strings, upper and lower case characters are to be treated identically.

Since the position of the system's name in the host's answer is unknown, it can not be returned in a buffer. If the application wants to have the full host's answer, it must either use the verbose mode (FtpsetVerboseMode) or use the FtpQuote command.

Syntax FtpSyst (LPSTR FAR *szSystStr)

Arguments: szSystStr An array of strings that contains the system names to be checked.

Return Codes:

The index of the array of strings	
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_SYSTUNKNOWN	The server has returned a string, but its answer does not match with the array of strings given as argument.
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDREFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREPLY	FTP4W has received no reply. FTP4W does not close the connection socket (use FtpLocalClose).
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer. FTP4W does not close the connection

Example:

```
static char *szSystem[] = { "Unix", "VMS", "Dos", NULL };

Rc=FtpSyst (szSystem);
printf ("System %s", Rc==FTPERR_SYSTUNKNOWN ? "Unknown": szSystem[Rc]);
```