

FTP4W API User Manual
Version 2.6

License

FTP4W was written by Philippe Jounin and is Copyrighted 1994-1996 by him and the SNCF (French Railways). The author disclaims all liability for its use or for problems, data corruption, data loss, or other loss that may result from its use.

Permission is given without restriction to use and distribute the program provided that it is distributed without charge, that it is not modified in any way, and that this file accompanies the DLL file.

This program may be included on CD-ROMs or other distribution methods freely, provided any charge for such is for recovering the cost of distribution and reasonable profit and not for the purpose of "selling" the program. In this case the distribution must contain the complete program including this file.

Send any comments to ark@ifh.sncf.fr.

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 with ToolBook)
- Richard Terpstra <terpstr2@ksla.nl> and Kees de Rooij (VB Sample)
- Vince Vielhaber <vev@msen.com> (Tests with VMS and C++)
- Bram Buitendijk <Bram.Buitendijk@library.KNAW.nl> (Tests with MS-Access)
- Teemu Mottonen <Teemu.Mottonen@ktl.fi> (Corrections of this manual)
- Robin Bowes <robin@plato.ucsf.ac.uk> (Tests with Paradox)
- 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>
- Michael Douglass <mikedoug@texas.net> (Ftp4w.hlp)
- Joachim Durchholz <durchjac@koa.ina.de> Tons of suggestions

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 mechanism 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 different 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)
- Dec Pathworks

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

Important changes

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

New functions :

14.03.95	Version 2.0	FtpPWD, FtpMKD, FtpRMD, FtpCDUP, FtpDelete
15.03.95	Version 2.0	Passive mode
27.03.95	Version 2.0	FtpSyst, FtpRenameFile
29.03.95	Version 2.0	FtpAppendToRemoteFile
07.04.95	Version 2.2	FtpAppendToLocalFile
29.08.95	Version 2.3	FtpSendAccount
16.11.95	Version 2.4	FtpRestartSendFile, FtpRestartRecvFile
08.02.96	Version 2.5	FtpOpenDataConnection, FtpCloseDataConnection, FtpSendThroughDataConn
20.02.96	Version 2.5	FtpFirewallLogin
25.02.96	Version 2.5	FtpErrorString
14.03.96	Version 2.5	FtpMGet
01.04.96	Version 2.5	FtpBufferPtr
10.06.96	Version 2.6	IntTnSend/IntFtpGetAnswerCode/FtpFirewallLogin

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 and its library file Ftp4w.Lib
- The DLL to be used with Lan Workplace FTP4WLWP.DLL
- The 32 bits version FTP4W32.DLL and its library file Ftp4w32.Lib
- A C-header file Ftp4w.h
- A Pascal header file UseFtp4w.Pas
- A Visual Basic header file Ftp4w.Bas

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.

Interpreting FTP4W errors

The FTP4W error codes live in the following ranges:

-1	Abortion of action in progress	FTPERR_CANCELBYUSER	Action aborted via FtpAbort.
0	Positive acknowledge, FTP4W and remote FTP server program have taken requested action.	FTPERR_OK	Successful function.
1-99	Positive acknowledge, but further information is required.	FTPERR_ENTERPASSWORD	FtpSendPasswd needed after FtpOpenConnection.
		FTPERR_ACCOUNTNEEDED	FtpSendAccount needed after FtpSendPasswd.
		FTPERR_RESTARTOK	Restart command successful.
100-599	Return codes from the remote FTP server program, passed through to the application program by FTP4W. These codes are defined in RFC 959. They are classified according to the following scheme:		
	100-199		Positive preliminary reply. FTP server has started requested action.
	200-299		Positive acknowledge. The FTP server has done what was requested, or the requested action was not necessary.
	300-399		Positive intermediate reply. FTP server needs a specific command now (e.g. a password after a FtpConnect).
	400-499		Negative reply, but error condition may go away without specific action. A retry of the command would make sense.
	500-599		Negative reply. The command will not work if tried again.
	x0x		Command has a syntax error. Non-functional, unimplemented or superfluous commands.
	x1x		Informational message. (E.g. after HELP.)
	x2x		Reply refers to status of a connection.
	x3x		Authentication and accounting replies.
	x5x		Reply refers to status of remote host's file system.
1000-1999	Programmer's and/or user's errors. These errors originate from FTP4W itself.	FTPERR_INVALIDPARAMETER	Parameter error.
		FTPERR_SESSIONUSED	Second call to FtpInIt in program, without intervening FtpRelease.
		FTPERR_NOTINITIALIZED	FtpInIt has not been called since start of program or since last FtpRelease.
		FTPERR_NOTCONNECTED	There was no successful FtpLogin or FtpOpenConnection since last FtpInIt.

the data connection for file transfer. (Data connections are created for each file or directory transfer, and they are destroyed immediately after the transfer is complete.)

FTPERR_TIMEOUT Timeout :
FTP4W has not been able to read data from the server (ie: the server has not sent data or the network is down).

FTPERR_FWCANCONNECT
 Networking error when connecting to the firewall

FTPERR_FWCONNECTREJECTED The firewall host rejects the connection from the client.

3000-3999 Error messages from the FTP server program, predigested by FTP4W.

FTPERR_UNEXPECTEDANSWER The FTP server sent a response that FTP4W didn't expect. FTP4W doesn't know whether the command was successful or not.

FTPERR_CANNOTCHANGETYPE The FTP server cannot change to the requested transfer type (ASCII, BINARY, or L8).

FTPERR_CMDNOTIMPLEMENTED The FTP server recognized the command, but it does not know how to execute it.

FTPERR_PWDBADFMT The FTP server reports its current working directory, but FTP4W cannot parse the answer.

FTPERR_PASVCMDNOTIMPL The FTP server does not support passive mode. (FTP4W needs this mode for several functions.)

5000-5999 Resource errors

FTPERR_CANTCREATEWINDOW
 Insufficient Windows resources for creating a window. (FTP4W creates an invisible window for handling the message traffic from WINSOCK.DLL.)

FTPERR_INSMEMORY Insufficient heap memory for FTP4W's internal data structures.

FTPERR_CANTCREATESOCKET
 Problem creating WINSOCK socket.

FTPERR_CANTBINDSOCKET
 Problem binding WINSOCK socket.

FTPERR_SYSTUNKNOWN FtpSyst doesn't recognize system name reported by FTP server (see FtpSyst).

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
FtpBufferPtr	Returns a pointer on the last string returned by the server
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 an FTP session
FtpCloseDataConnection	Closes an FTP connection for a file transfert
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
FtpMGet	Get all files from a remote directory
FtpMKD	Creates a remote directory
FtpOpenConnection	Makes an FTP connection
FtpOpenDataConnection	Makes an FTP connection for a file transfert
FtpPWD	Gets the remote default directory
FtpQuote	Sends a user-defined command to the server
FtpRecvFile	Retrieves a remote file
FtpRecvThroughDataConnection	Reads data from the data channel
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
FtpSendThroughDataConnection	Inserts data on the data channel
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

I am still working on a function which will enables Ftp4w to work through a firewall. If you are intersted, please contact me, i will send you the code that i have written.

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.

FtpAppendToLocalFile:

- FTPERR_NOTINITIALIZED
- FTPERR_NOTCONNECTED
- ToolsSetDataConnection()
- FTPERR_OK
- FtpSyncRecv ()

FtpAppendToRemoteFile:

- FTPERR_NOTINITIALIZED
- FTPERR_NOTCONNECTED
- ToolsSetDataConnection
- FTPERR_OK
- FtpSyncSend

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:

FALSE
TRUE

Error
FtpAbort has been done

FtpBufferPtr

This function returns a pointer on the last string returned by the server. As described above, this string begins by three digits which compose the server's return code, and a free text which gives more information.

Possibly, this command can be used to display an error message after the reception of a negative code (ie: FTPERR_CANTOPENFILE).

Syntax: FtpBufferPtr()

Return codes : A 32 bit pointer to the last string returned by the server.

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.

FtpBytesToBeTransferred returns the total length of the file which is transferred. 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 *Transferred* instead of *Transferred*, the functions *FtpBytesTransferred* and *FtpBytesTransferred* are still implemented on the 16 bits version for backwards compatibility.

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
FTPERR_FILELOCKED	
FTPERR_CMDNOTIMPLEMENTED	

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_SENDFUSED	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_NOTCONNECTED	

FtpCloseDataConnection

This function tries to close the data connection.

This function is reserved for experimented programmers. It must be used in conjunction with FtpOpenDataConnection.

Syntax: FtpCloseDataConnection (void)

Return Codes

FTPERR_OK	FTP session has been closed
FTPERR_CANTCLOSE	Error during close
FTPERR_CANTWRITE	The server has not written all data sent by the client
FTPERR_UNEXPECTEDANSWER	FTP4W has received a reply. But this reply is not a valid FTP answer.

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_SENDFUSED	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_FILELOCKED	<jhd>
FTPERR_CMDNOTIMPLEMENTED	<jhd>

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_SENDFUSED	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_CMDNOTIMPLEMENTED	

FtpDir

This function reads the remote directory.

* Asynchronous Mode

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.

* Synchronous Mode

The application must specify a file name, which will be filled with the remote directory. The function 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 wParam);

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.
wParam	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_SENDFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server rejects the command TYPE ASCII
FTPERR_CANTOPENLOCALFILE	Local file can not be 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
FTPERR_INVALIDPARAMETER	
FTPERR_NOREMOTEFILE	Remote directory is empty
FTPERR_CANCELBYUSER	FtpAbort has been called
FTPERR_DATACONNECTION	Network Error
FTPERR_CMDNOTIMPLEMENTED	Server does not understand the TYPE command.
FTPERR_CANNOTCHANGETYPE	The server has rejected the command TYPE

FtpFirewallLogin

Note: This function is not implemented since i have no way to test it. Please contact me in order to receive the source code of this function.

This function tries to establish a connection with an remote host through a firewall.

The nTypeFirewall precises which firewall type is used:

FTP4W_FWSITE	remote host is reached by SITE command firewall asks for USER/PASSWD
FTP4W_FWPROXY	remote host is reached by OPEN command firewall does not asks for USER/PASSWD
FTP4W_FWUSERWITHLOGON	remote host is reached by USER@HOST command and firewall asks for USER/PASSWD
FTP4W_FWUSERNOLOGON	remote host is reached by USER@HOST command and firewall does not ask for USER/PASSWD

Syntax : int PASCAL FAR FtpFirewallLogin(
 LPSTR szFWHost, LPSTR szFWUser, LPSTR szFWPass,
 LPSTR szRemHost,LPSTR szRemUser,LPSTR szRemPass,
 int nFirewallType,
 HWND hParentWnd, UINT wMsg)

Arguments:

szFWHost	The name of the firewall
szFWUser	The name of the user on the firewall
szFWPasswd	The passwd on the firewall
szRemHost	The name to the FTP server
szRemUser	The name of the user for FTP server
szRemPasswd	The Password of the user for FTP server
nFirewallType	Identifies the firewall type
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	User is logged on
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_INVALIDPARAMETER	Required parameters are NULL or nTypeFirewall is incorrect
FTPERR_SENDREFUSED	Error happens during send
FTPERR_FWCANTCONNECT	Error during connection with firewall
FTPERR_FWCONNECTREJECTED	Firewall rejects log on command
FTPERR_FWLOGINREFUSED	Login on firewall refused
FTPERR_CANTCONNECT	FTP server unreachable/unknown...
FTPERR_LOGINREFUSED	Login on FTP server refused

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 immediately 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
FTPERR_WINSOCKNOTUSABLE	Ftp4w needs a Winsock version 1.1 or higher

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:

FTPERR_OK
FTPERR_NOTINITIALIZED
FTPERR_CANTCLOSE

FtpLogin

This function combines the three functions FtpOpenConnection, FtpSendUserName, and FtpSendPassword.

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 immediately returns FTPERR_OK, and then later, when the request is completed, the application will receive a wParam message in the hWnd window.

The message will be followed by:

wParam: TRUE

lParam: The return code of the function

Syntax:

FtpLogin(LPCTSTR szHost,LPCTSTR szUser,LPCTSTR szPass, HWND hWnd,WMSG wParam)

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

wParam 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_SENDFUSED	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
FTPERR_INVALIDPARAMETER	Bad parameters

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.

FtpMGet

This function gets specified files from the remote server.

This function needs an user-defined function which is to be called twice with 3 parameters :

- The name of the remote file which should be transferred or which has been transferred
- The name of the local file which should be created or which has been created
- an int value which is set to FTPERR_ENDOFDATA if the file transfer is

- Before a file transfer

the function can return FALSE to skip this transfer or change the name of the files

The user-defined function can change the name of the file or decide to skip this file by returning FALSE.

Notes :

This command should be used in synchronous mode only.

if szFilter is not NULL, the behaviour of FtpMGet depends on the FTP server.

Syntax: FtpMGet (LPSTR szFilter, char cType, BOOL bNotify, BOOL FAR (*f) ...

Argument: szFilter The selection mask. It can include wilcards ("*", "**.*") or it can be a directory name. If this parameter is set to NULL all parameters from the current directory will be copied

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.

f NULL or a user-defined function which accepts three arguments :

- szRemFile a 256 bytes zone which contains the name if the file which is to be received.
- szLocalFile a 256 bytes zone which contains the name of the file which will be created (initailized to szRemFile)
- The last return code if the transfer has been attempted or FTPERR_ENDOFDATA if the file is to be transferred

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 successfully 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_CMDNOTIMPLEMENTED	<jhd>
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_UNKNOWNHOST	The hostname can not be resolved
FTPERR_CANCELBYUSER	User has called FtpAbort
FTPERR_SERVERCANTEXECUTE	The remote FTP server can not accept the connection : too many users logged on, server off-line, ...
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.

FtpOpenDataConnection

This function is reserved for experimented programmers which needs to control the data sent on the data streams. It must be used in conjunction with FtpCloseDataConnection and FtpRecvThroughDataConnection or FtpSendThroughDataConnection.

It opens the data connection with the server on port 20. It does not send any data through this channel nor wait for them.

By default the active mode is used (ie. Ftp4w opened a listened socket and waits for a connection), although this mode may be changed by FtpSetPassiveMode.

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

The action to be executed by the server is given in nAction. It should be either

FTP4W_STORE_ON_SERVER	The data sent by the client are written in the file szRemote
FTP4W_APPEND_ON_SERVER	The data sent by the client are written at the end of the file szRemote
FTP4W_GET_FROM_SERVER	The server sends the contents of the file szRemote

Syntax: FtpOpenDataConnection (LPSTR szRemote, int nAction, char cType)

Argument: szRemote The remote file to be written or to be read.
nMode The action to be executed by the server
cType The data representation (ASCII, IMAGE, L 8)

Return codes:

FTPERR_OK	Successful connection
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_PASVCMNDNOTIMPL	Server does not support Passive mode
FTPERR_SENDREFUSED	ftp4w can not send data
FTPERR_CANTCREATESOCKET	The socket has not been created
FTPERR_DATACONNECTION	Error during connection
FTPERR_CANCELBYUSER	Ftp4w has received an abort command
FTERR_TRANSFERREFUSED	Transfer is refused by the server
FTPERR_NOREMOTEFILE	File to be read has not been found
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:

	Name of the remote directory
FTPERR_OK	available in the buffer
FTPERR_SERVERCANTEXECUTE	PWD has failed (directory does not exist,...)
FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	User is not connected to a remote host
FTPERR_SENDFUSED	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
FTPERR_CMDNOTIMPLEMENTED	Command is not implemented

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_SENDFUSED, ...).

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:

FTPERR_NOTINITIALIZED	Session has not been initialized by FtpInit
FTPERR_NOTCONNECTED	
FTPERR_SENDFUSED	Ftp4w can not send data
FTPERR_NOREPLY	

A 3 digits number between 100 and 699.

FtpRecvFile / FtpAppendToLocalFile

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 overwrites it.

This function copies a remote file to a local file. In the asynchronous mode, the function returns immediately, 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 hWnd 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 exist, it is created in any case.

Syntax:

FtpRecvFile (LPSTR szRemote, LPSTR szLocal,
char cType, BOOL bNotify,
HWND hWnd, UINT wMsg)

FtpAppendToLocalFile (LPSTR szRemote, LPSTR szLocal,
char cType, BOOL bNotify,
HWND hWnd, 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_PASVCMNDNOTIMPL	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_SENDFUSED	FTP4W can not send the data (network is down)
FTPERR_NOREMOTEFILE	Remote file has not been found
FTPERR_CANCELBYUSER	Operation aborted by FtpAbort
FTPERR_DATACONNECTION	Error in data connection
FTPERR_SERVERCANTEXECUTE	Server rejects the command
FTPERR_CANNOTCHANGETYPE	The server has rejected the command TYPE
FTPERR_CANTOPENLOCALFILE	Local file can not be 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

FtpRecvThroughDataConnection

This command allows the application to reads data from the file transfer stream. It should be used if the application needs to process in "real time" the incoming data.

The data connection should have been opened by FtpOpenDataConnection.

The application should call FtpCloseDataConnection after receiving an FTPERR_ENDOFDATA return code.

Syntax: int FtpRecvThroughDataConnection (LPSTR szBuf, unsigned int *lpBufSize)

Arguments: szBuf The buffer which will receive the data
lpBufSize A pointer on an integer which gives the size of szBuf and which will be set to the amount of read data.

Return Codes:

FTPERR_OK	Data has been received
FTPERR_ENDOFDATA	The server has no more data to send
FTPERR_DATACONNECTION	Error during recv or time-out.
FTPERR_CANCELBYUSER	Transfer aborted by FtpAbort

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.
FTPERR_NOACTIVESESSION	FtpInit has not been called

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 IByteTransfer)

Argument : IByteTransfer 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 wParam);

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
wParam	the application-defined message to pass to the application

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

Return Codes:

FTPERR_OK	File received
FTPERR_PASVCMNDNOTIMPL	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_SENDFREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server has rejected the command TYPE
FTPERR_CANTOPENLOCALFILE	Local file can not be 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.
FTPERR_CMDNOTIMPLEMENTED	Command not implemented on the remote server
FTPERR_NOREMOTEFILE	File has not been found
FTPERR_CANCELBYUSER	Transfer aborted
FTPERR_DATACONNECTION	Error in data connection
FTPERR_SERVERCANTEXECUTE	Command rejected by the server

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_PASVCMNDNOTIMPL	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_SENDFREFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server rejects the command TYPE ASCII
FTPERR_CANTOPENLOCALFILE	Local file can not be 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
FTPERR_CMDNOTIMPLEMENTED	Command not implemented
FTPERR_CANCELBYUSER	Transfer aborted
FTPERR_DATACONNECTION	Error in data connection
FTPERR_SERVERCANTEXECUTE	Command rejected by the server

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
FTPERR_FILELOCKED	
FTPERR_CMDNOTIMPLEMENTED	

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_SENDFUSED	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_CMDNOTIMPLEMENTED	

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 overwrites 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 wParam) are not used.

Syntax:

FtpSendFile (LPSTR szLocal, LPSTR szRemote,
char cType, BOOL bNotify,
HWND hParentWnd, UINT wParam)

FtpAppendToRemoteFile (LPSTR szLocal, LPSTR szRemote,
char cType, BOOL bNotify,
HWND hParentWnd, UINT wParam)

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
	wParam	the application-defined message to pass to the application

Return Codes:

FTPERR_OK	File has been sent
FTPERR_PASVCMNDNOTIMPL	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_SENDFUSED	FTP4W can not send the data (network is down)
FTPERR_CANNOTCHANGETYPE	The server rejects the command TYPE ASCII
FTPERR_CANTOPENLOCALFILE	Local file can not be 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

FTPERR_CANCELBYUSER	Transfer aborted
FTPERR_DATACONNECTION	Error in data connection
FTPERR_SERVERCANTEXECUTE	Command rejected by the server

FtpSendThroughDataConnection

This command allows the application to send data by the file transfer stream. It should be used if the data to be sent do not come from a file.

To end the data transfer, the application should call `FtpCloseDataConnection`.

Syntax: `int FtpSendThroughDataConnection (LPSTR szBuf, unsigned int uBufSize)`

Arguments: `szBuf` The buffer which contains the data
`uBufSize` The amount of data to be sent

Return Codes:

<code>FTPERR_OK</code>	Data has been received
<code>FTPERR_DATACONNECTION</code>	Error during recv or time-out.

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_ACCOUNTNEEDED	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_SENDFUSED	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_CMDNOTIMPLEMENTED	

FtpSendUserName

This function sends the user's name to the server. This authentication 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_SENDFUSED	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_SERVERCANTEXECUTE	Command rejected by the server

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 port 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)
```

This 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.

This 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.

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
FTPERR_CMDNOTIMPLEMENTED	
FTPERR_CANNOTCHANGETYPE	

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 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:

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

Return codes

<code>FTPERR_OK</code>	Mode has been changed
<code>FTPERR_NOTINITIALIZED</code>	session has not been initialized by <code>FtpInit</code>

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_SENDFUSED, ...). 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_SENDFUSED	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_CMDNOTIMPLEMENTED	<jhd>

Example:

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

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