FTP4W API User Manual

Licence

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

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. The functions return an integer which is the 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 parameters 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, 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 calls.

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 parameters :

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

The FTP4W calls do not need any handle to identify the FTP session. They use the Windows function **GetCurrentTask** to get a task identifier. This mechanisms avoids the use of a parmeter 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).

Programming with the FTP4W API

To use the FTP4W functions 4 files are provided :

- This reference
- The DLL FTP4W.DLL
- An include file FTP4W.H
- 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.

The task is ready to make a connection with a FTP server. Il 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 (i.e. the network has been shutdown), it must call **FtpLocalClose**.

To release the allocated buffers, the application must call **FtpRelease** before it exits.

The FTP4W functions

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.

The function needs the handler of an application window or NULL if any.

Syntax : FtpInit (HWND hParentWnd)

parameters : 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

FtpDataPtr

FtpDataPtr returns the address of the internal structure LPProcData.
Its use should be made only by the macros listed below.

FtpSetVerboseMode

If a programmer wants to have a look on each frame sent by the server, he must use this function. He will get a message (by the **SendMessage** function) each time a frame has been received. The parameter wParam is TRUE, lParam points to the frame. It is nul terminated but can contains 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)

parameters :	bVerboseMode	TRUE if the application wants to control 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_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit

FtpSetSynchronousMode / FtpSetAsynchronousMode / FtpIsAsynchronousMode

Syntax :

```
#define  FtpSetAsynchronousMode()   FtpDataPtr()->File.bAsyncMode=TRUE
#define  FtpSetSynchronousMode()    FtpDataPtr()->File.bAsyncMode=FALSE
#define  FtpIsAsynchronousMode()    (FtpDataPtr()->File.bAsyncMode)
```

These macros allow to change the mode of data transfer calls.
By default the asynchronous mode is used.

FtpIsAsynchronousMode returns a boolean which is TRUE if the current mode is the asynchronous one.

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

FtpSetNewDelay / FtpSetNewSlices

Syntax :

```
#define  FtpSetNewDelay(x)         FtpDataPtr()->File.nDelay=x
#define  FtpSetNewSlices(x,y)      FtpDataPtr()->File.nAsyncAlone=x,
                                   FtpDataPtr()->File.nAsyncAlone=y,
```

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 parameter 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 parameter is the number of frames when one FTP session are active, the second parameter is used when two or more sessions are active.
Both parameters should not be set to zero.

FtpSetDefaultPort / FtpSetDefaultTimeOut

Syntax :

```
#define FtpSetDefaultPort(x)        FtpDataPtr(x)->ftp.nPort=x
#define FtpSetDefaultTimeOut(x)     FtpDataPtr(x)->ftp.nTimeOut=x
```

These macros are used to change either the FTP-control port (21 by default) or the timeout (30 secondes by default).

The new timeout is given in seconds.

FtpBytesTransfered / FtpBytesToBeTransfered

Syntax :
```
#define  FtpBytesTransfered()          FtpDataPtr()->File.lPos
#define  FtpBytesToBeTransfered()   FtpDataPtr()->File.lTotal
```

FtpBytesTransfered returns the number of bytes which has been transfered. This number is reset for each new transfer.

FtpBytesToBeTransfered returns the total length of the file which is transfered. For ASCII transfers, it can be slighty different from the number of bytes to be received. Furthermore, if the result of this macro is 0, it means that FTP4W has not been able to get this information.

FtpOpenConnection

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

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

FTP4W does not verify if a connection has already been done.

Syntax : FtpOpenConnection (LPSTR szHost)

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

Return codes :

FTPERR_OK	Successful connection
FTPERR_SESSIONNOTINITIALIZED	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.

FtpSendUserName

This function sends the user's name to the server. This authentification is necessary to begin a file transfer.

Syntax : FtpSendUserName (LPSTR szUserName)

Parameter : szUserName : Name of the user

Return Codes :

FTPERR_OK	User is logged on
FTPERR_ENTERPASSWORD	Successful function but server awaits a password.
FTPERR_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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.

FtpSendPasswd

This function sends the password to the server.

Syntax : FtpSendUserName (LPSTR szUserName)

Parameter : szUserName : Name of the user

Return Codes :

FTPERR_OK	User is logged on
FTPERR_ENTERACCOUNT	Successful function but server awaits a account name
FTPERR_LOGINREFUSED	The USER/PASSWD has been rejected
FTPERR_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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.

FtpLogin

This function combines the three preceding functions. It completes the login procedure.
If the current mode is the synchronous mode, FtpLogin will return when the job is over, the two last parameters are unused. Otherwise it returns FTPERR_OK, once the request has been 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)

Parameters : 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 parameter:

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_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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

FtpCloseConnection

This function try to close gracefully the connection.It will not succeed if a file transfer is in progress or if the server has timed-out. You must then use FtpLocalClose.

Syntax : FtpCloseConnection (void)

Return Codes

FTPERR_OK	FTP session has been closed
FTPERR_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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.

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 :
return FALSE if the session has not been initialized by FtpInit
else return TRUE.

FtpCWD

This function changes the default directory on the remote server.

Syntax : FtpCWD (LPSTR szPath)

Parameter : szPath : name of the new directory

Return Codes :

FTPERR_OK	Directory has been changed
FTPERR_SERVERCANTEXECUTE	CWD has failed (directory does not exists..)
FTPERR_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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

FtpSetType

This function changes the default transfer type.

Syntax : FtpSetType (char cType)

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

Return Codes :

FTPERR_OK	type has been changed
FTPERR_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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

FtpAbort

This function aborts a data transfer without breaking the connection.
This function returns immediatly. The data transfer is really aborted when the 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

FtpDir

This function reads the remote directory.

* Asynchronous Mode

It can be used in 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 nul-terminated and contains only one line (the ending <CR><LF> has been removed). The last 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 functions returns an error code or FTPERR_OK if it is successful.
The two last parameters are ignored.

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

Parameters : 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_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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

FtpRecvFile

This function copies a remote file to a local file. In the asynchronous mode, the funvction 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 last parameters are not used.

Syntax :
FtpRecvFile (LPSTR szRemote, LPSTR szLocal,
char cType, BOOL bNotify,
HWND hParentWnd, UINT wMsg)

Parameters : szRemote Remote file specification
szLocal The file where to write the data.
cType TYPE_A for ASCII, TYPE_B for binary
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_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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 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

FtpSendFile

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 when the transfer is 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 last parameters are not used.

Syntax :

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

Parameters : szLocal	The file to be sent	
	szRemote	Remote file specification
	cType	TYPE_A for ASCII, TYPE_B for binary
	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	Dir has been done
FTPERR_SESSIONNOTINITIALIZED	session has not been initialized by FtpInit
FTPERR_CANTSEND	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

FtpGetFileSize

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 last reply if the server has sent the size of the file.

If the function is successful, it will return the length of the file. (Note that in ASCII mode, it can be slighty different from the number of bytes FTP4W will receive). else it returns 0.

This function is obsolete and should be replaced by the macro FtpBytesToBeTransfered.

Syntax
FtpGetFileSize()

returns DWORD.

FtpQuote

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

The return code is either a FTP code (ie 200) or a FTP4W error code (ie FTPERR_CANTSEND). The reply (if any) is copied into a user's buffer.

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

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