Visual Foxpro 5.0+ FTP Service Class

Written by: Robert Abram
Rev Date: 5/99

Disclaimer

The Program Code and Documentation given for Visual Foxpro FTP Service is provided *as is* with no warranty, either implied or expressed. The author assumes no responsibility for any damages or loss of information caused through the use or misuse of this Visual Foxpro FTP Service code.

The FTP Service Program Code and Documentation is free to all persons. The FTP Service Code can be freely distributed in any commercial or private software package. The FTP Service code can also be modified for special use in an application. No person or persons should distribute modified FTP Service Code.

The Author requests a note of acknowlegement as a contributer in any software package this code is used in.

Overview

The code written here has been written as a Visual Foxpro Class. The FTP Class provides a set of functions that allow FTP service to be added to a program with little programmer overhead.

Note: Wininet.dll is required to be available on any computer these functions are to be used on. Wininet.dll is a product of the Microsoft Corporation. Microsoft allows the Wininet.dll to be bundled with any application using the Wininet.dll internet service functions. The Wininet.dll is installed if IE 4.0+ has been installed on the computer and it is also installed as part of newer versions of Microsoft Windows.

The FTP Class is completely stand alone, other than Wininet.dll, and completely written in the Foxpro language. Giving an application programmer easy access to the functionality provided by the class.

How to get started

To get your program access to the class, the following code should be inserted somewhere in your code before calling the FTP Service functions;

```
Set Procedure To ftp.prg Additive
```

This tells Foxpro where to look for the FTP Class. The next thing to do is to create an instance of the FTP Class. You need to add code something like this;

```
sz_ftp = createobject('ftp_service')
```

The new object is held in the "sz_ftp" variable and should be referenced by this variable. For example;

```
sz_ftp.OpenInternet("user", "password", "ftpaddress", "port")
```

Important Program Notes

All file transfers are done in Binary mode. Ascii and Binary are available, but after looking at the differences I decided that Binary is the only mode I would use for transfering files anyway. If Ascii is necessary, the Class would need to be modified to allow selection of the Ascii transfer method.

Before this documentation, I tested the FTP class pretty throughly. During the writing of this documentation, I decided to make changes to some of the code. So if there are some inconsistancies or errors, it comes as no suprise and falls under the disclaimer. But, since I am such a great programmer (ha), there shouldn't be any problems (ha ha). If you do find something needing fixing or have a question, you can post a message on the Universal Thread or contact me on the Universal Thread about the problem.

Additional links for more information

http://www.microsoft.com/workshop/networking/wininet/reference/reference.asp

File Caching when downloading

5/99 - I have added some caching support for file downloading. There are two new functions to change the file caching when downloading a file. **SetCachingMethod** and **GetCachingMethod**. The Default is No File Caching.

Some people have had problems setting up auto Ftp programs that retrieve the save file on a recurring basis. This was because, after the first the time file was downloaded, the file was put into cache and subsequent calls for the file retrieved the file from the cache and not the FTP server. The file date was set to the time of the latest download attempt, adding to the confusion.

File Download Caching Functions

SetCachingMethod

Function Syntax:

```
SetCachingMethod (
                 Int NewCacheMethod )
```

Parameters:

NewCacheMethod - Can be on of the following

```
INTERNET_FLAG_DONT_CACHE
INTERNET_FLAG_HYPERLINK
INTERNET_FLAG_MAKE_PERSISTENT
INTERNET_FLAG_MUST_CACHE_REQUEST
INTERNET_FLAG_RELOAD
INTERNET_FLAG_NO_CACHE_WRITE
INTERNET_FLAG_RESYNCHRONIZE
```

Returns:

True if successful

Comments:

Changes the current file caching method. Subsequent calls to GetFtpFile will use the new file caching method.

GetCachingMethod

Function Syntax:

GetCachingMethod ( )

Parameters:

None

Returns:

Integer – The current caching method.

Comments:

Returns the current caching method.

Dialup Connection Functions

AutoDialInternet

Function Syntax:

```
AutoDialInternet(
        Int ConnectType
 )
```

Parameters:

ConnectType - Can be one of the following;

INTERNET_AUTODIAL_FORCE_ONLINE,
INTERNET_AUTODIAL_FORCE_UNATTENDED,
INTERNET_AUTODIAL_FAILIFSECURITYCHECK

Returns:

True if successful

Comments:

Causes the modem to automatically dial the default Internet connection.

AutoDialHangUp

Function Syntax:

```
AutoDialHangUp ( )
```

Parameters:

None

Returns:

True if successful

Comments:

Closes a modem connection that was created from an AutoDialInternet function call.

DialInternet

Function Syntax:

```
DialInternet (
        char DialUpName,
        int ConnectType
 )
```

Parameters:

DialUpName – Name of the DialUp Connection to use.
ConnectType - Can be one of the following;
INTERNET_AUTODIAL_FORCE_ONLINE,
INTERNET_AUTODIAL_FORCE_UNATTENDED,
INTERNET_AUTODIAL_FAILIFSECURITYCHECK

Returns:

Integer – Connection handle number. 0 if API Call unsuccessful or –1 if foxpro function failed.

Comments:

Attempts to use a preconfigured Dial Up account to connect to the internet.

DialHangUp

Function Syntax:

```
DialHangUp (
        Int ConnectionHandle
 )
```

Parameters:

ConnectionHandle – Handle returned from the DialInternet Function Call.

Returns:

True if successful.

Comments:

Closes a dialup connection that was created from a DialInternet function call.

GetConnectedState

Function Syntax:

GetConnectedState ( )

Parameters:

None

Returns:

Integer – Current Connection Type; INTERNET_CONNECTION_MODEM, INTERNET_CONNECTION_LAN, INTERNET_CONNECTION_PROXY, INTERNET_CONNECTION_MODEM_BUSY. Possibly other codes could be returned.

Comments:

Returns the current type of internet connection..

GoOnline

Function Syntax:

```
GoOnline (
        string URL,
 )
```

Parameters:

URL – The URL program is requesting to go to.

Returns:

True if successful

Comments:

Brings up a dialog to the user requesting permission to go the requested URL.

FTP Service Functions

Note: This list is the relevant functions in the FTP Class. Other functions in the Class are used internally by functions listed below.

OpenInternet

Function Syntax:

```
OpenInternet (
        char UserName,
        char UserPassword,
        char FTP Address,
        char FTP Port
)
```

Parameters:

UserName	- A valid user id on the FTP server.
UserPassword	- A valid password for the UserName.
FTP Address	- The Site Alias or IP number of the FTP Server.
FTP Port	- The number of the Port on the Server to connect to.

Returns:

Returns .T. if the function succeded in making a connection to the internet and the FTP server. Returns .F. if the function was unable to connect to the internet or the FTP server.

Comments:

Note: Port 21 is the default FTP port used on FTP servers.

Attempts to open a connection to the FTP server. If successful, the function queries the directory the server placed the connection to. This function must be called before any other FTP function.

CloseInternet

Function Syntax:

```
CloseInternet ( )
```

Parameters:

None

Returns:

Nothing

Comments:

Closes any open handles to the Internet. Does not cause an error if there are no currently open handles.

GetFtpFile

Function Syntax:

```
GetFtpFile (
        char RemoteFile,
        char LocalFile,
        logical FailifExist
)
```

Parameters:

RemoteFile	- Name and Path of the file on the FTP Server to bring down.
LocalFile	- Name and Path for the contents of the file to be stored in.
FailifExist	- .T. to fail if the local file exists. .F. to overwrite the local file.

Returns:

Returns .T. if the function successfully copied the file from the FTP Server.
Returns .F. if the operation failed.

Comments:

This function will bring a file from the FTP Server down to a location specified in LocalFile.

PutFtpFile

Function Syntax:

```
PutFtpFile (
        char RemoteFile,
        char LocalFile
)
```

Parameters:

RemoteFile	- Name and Path for the contents of the file to be stored in.
LocalFile	- Name and Path of the file to send to the FTP Server.

Returns:

Returns .T. if the function successfully copied the file to the FTP Server.
Returns .F. if the operation failed.

Comments:

This function sends a file to the FTP server from some local area. Depending on how the FTP Server is setup, decides whether or not an existing file is overwritten or if the function fails to send the file.

DeleteFtpFile

Function Syntax:

```
CloseInternet (
        Char RemoteFile
 )
```

Parameters:

RemoteFile	- Name and Path of the File to delete on the FTP Server.

Returns:

Returns .T. if the function successfully deleted the file from the FTP Server.
Returns .F. if the function could not delete the file.

Comments:

This function can only delete files off of the FTP Server that are not read only or some how protected by the server.

RenameFtpFile

Function Syntax:

```
RenameFtpFile (
        char OldName,
        char NewName
)
```

Parameters:

OldName	- Current file name on FTP Server.
NewName	- New name for file on FTP Server.

Returns:

Returns .T. if the function successfully renamed the file on the FTP Server.
Returns .F. if the function could not rename the file.

Comments:

Pretty simple, just renames a file.

CreateFtpDirectory

Function Syntax:

```
CreateFtpDirectory (
        char Directory
)
```

Parameters:

Directory	- Name of the directory to create on the FTP Server.

Returns:

Returns .T. if the function successfully created the directory on the FTP Server.
Returns .F. if the function could not create the directory on the server.

Comments:

Creates a directory on the FTP Server.

RemoveFtpDirectory

Function Syntax:

```
RemoveFtpDirectory (
        char Directory
)
```

Parameters:

Directory	- Name of the directory to remove on the FTP Server.

Returns:

Returns .T. if the function successfully removed the directory on the FTP Server. Returns .F. if the function could not remove the directory on the server.

Comments:

Removes a directory on the FTP Server. Probably, if the directory is not empty, The FTP Server will not remove the directory.

ChangeFtpDirectory

Function Syntax:

```
ChangeFtpDirectory (
        char Path
)
```

Parameters:

Path	- Name of the Path to change the current directory to.

Returns:

Returns .T. if the function successfully changed the current directory on the FTP Server. Returns .F. if the function could not change the directory.

Comments:

Tries to change the current directory on the FTP Server to the path specified.

GetFtpDirectory

Function Syntax:

```
GetFtpDirectory (
        char @Directory
)
```

Parameters:

Directory	- A pointer to a variable to hold the FTP Server directory.

Returns:

Returns .T. if the function successfully returned the current directory on the FTP Server. Returns .F. if the function could get the current directory.

Comments:

This function places the current directory on the FTP Server into the Directory Variable that was passed by reference to this function.

GetFtpDirectoryArray

Function Syntax:

```
GetFtpDirectoryArray (
        array @Files,
        char Mask
)
```

Parameters:

Files	- Holds an array of files after the function completes.
Mask	- Specifies a file mask to filter files in the operation.

Returns:

Returns .T. if the function successfully returned the current directory on the FTP Server. Returns .F. if the function could not get the current directory.

Comments:

This function retrieves a list of files from the current directory on the FTP Server. The files are placed in an array with each row being a file and that file's information. The array is created from a variable that is passed by reference to the function. So when the function returns, the Files variable contains an array of files.

Not all the array elements may be filled in, depending on the type operating system the FTP Server is running on. For example, if the FTP Server is running on NT Server, long file names are supported.

Note: After the function returns, doing an EMPTY(Files) will return .T. if no files were found in the directory on the FTP Server.

The Files array has the following structure;

Type	Pos	Name	Desc
C	[x, 1] =	FileName	- Long file name if available
C	[x, 2] =	Alternate FileName	- Short file name if available
N	[x, 3] =	File Size	- Size of file in bytes
T	[x, 4] =	File Create Date	- File create date and time
T	[x, 5] =	File Last Access Time	- Last access date and time
T	[x, 6] =	File Last Write Time	- Last write date and time
C	[x, 7] =	File Attributes	- file attributes

The file Attributes array index is a string with the following characters concatenated together;

'R'	FILE_ATTRIBUTE_READONLY
'H'	FILE_ATTRIBUTE_HIDDEN
'S'	FILE_ATTRIBUTE_SYSTEM
'D'	FILE_ATTRIBUTE_DIRECTORY
'A'	FILE_ATTRIBUTE_ARCHIVE
'N'	FILE_ATTRIBUTE_NORMAL
'T'	FILE_ATTRIBUTE_TEMPORARY
'C'	FILE_ATTRIBUTE_COMPRESSED
'O'	FILE_ATTRIBUTE_OFFLINE

GetErrorCode

Function Syntax:

```
GetErrorCode (
        logical Verbal
)
```

Parameters:

Verbal	- Set to .T. to display a MessageBox on the screen.

Returns:

Returns the error code from the last FTP function as a numeric value.

Comments

This function is the watch dog of this FTP Service. After every FTP function is performed, return codes, error codes and messages are stored internally. This function retrieves the last error code generated by a FTP function call.

Extended error codes and messages are generated during each API call. This extended information is also stored after each function call.

If Verbal is set to .T., a message box will be displayed showing the last error code with the text relating to that error code. Also the extended error code and message is displayed.

Most of the time, the error_code will be 0 if the function succeded.

In some cases, the error_code will be different than 0 if the function succeded. For example, this function returns ERROR_NO_MORE_FILES after the function GetFtpDirectoryArray succeds. So it is important to check to see if the function succeded and then check the error codes.

GetExtendedErrorCode

Function Syntax:

```
GetExtendedErrorCode ( )
```

Parameters:

None

Returns:

Returns an extended error code.

Comments

This function returns the extended error code that was generated during the last function call.

GetExtendedErrorMsg

Function Syntax:

GetExtendedErrorMsg ( )

Parameters:

None

Returns:

Returns the extended error message

Comments

Returns the extended error message generated during a function call.

GetErrorText

Function Syntax:

```
GetErrorText (
        numeric Error
)
```

Parameters:

Error	- An error code returned by GetErrorCode()

Returns:

Returns a text message relating to the error code passed.

Comments

This function is used internally, but seems like it might be useful as an exposed function. If a unknown error code is passed, this function just returns "Unknown Error Message"