FTP Client Engine
Library for Visual FoxPro

Programmer's Manual

(FCE4FP)

Version 3.4

November 17, 2015

This software is provided as-is.
There are no warranties, expressed or implied.

MarshallSoft Computing, Inc.
Post Office Box 4543
Huntsville AL 35815

Email: info@marshallsoft.com
Web: www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **FTP Client Engine for Visual FoxPro (FCE4FP**) library is a toolkit that allows software developers to quickly develop FTP client applications in Visual FoxPro.

The **FTP Client Engine (FCE)** is a DLL library that uses the Windows API to provide direct and simple control of the FTP protocol. The FCE component library can be used for both anonymous and private FTP sessions.

A straightforward interface provides the capability to easily build Visual FoxPro FTP software applications to connect to any FTP server, navigate its directory structure, list files, upload files, rename files, delete files, append files, and download files using the FTP protocol.

The **FTP Client Engine Programmers Manual** provides information need to compile and run programs in a Visual FoxPro programming environment.

The **FTP Client Engine for Visual FoxPro** supports all versions of 32-bit Visual FoxPro. **FCE4FP** includes multiple example programs that demonstrate FTP processing to help software developers easily build software applications using FCE4FP.

FCE4FP runs under all versions of 32-bit and 64-bit Windows through Windows 10. The **FTP Client Engine SDK** DLL's (FCE32.DLL and FCE64.DLL) can also be used with any development environment (Visual Basic, C++, Delphi, Visual FoxPro, COBOL, Xbase++, dBase, PowerBASIC, etc.) capable of calling the Windows API.

When comparing **FTP Client Engine** against our competition, note that:

- FCE4FP is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
- FCE4FP does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
- The FCE functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the FTP Client Engine Library for C/C++ (FCE4C), Delphi (FCE4D), PowerBASIC (FCE4PB), Visual Basic (FCE4VB), Visual dBASE (FCE4DB), and Xbase++ (FCE4XB). All versions of FCE use the same DLLs (FCE32.DLL or FCE64.DLL).

All versions of the FTP Client Engine Library (FCE) can be downloaded from our web site at

http://www.marshallsoft.com/ftp-client-library.htm

1.1 Features

Some of the many features of the **FTP Client Engine** component library are as follows:

- Works with both 32-bit and 64-bit Windows.
- Connect to any (anonymous or private) remote FTP server.
- Get a list of files (names or long format) on the server.
- Navigate the server directories.
- Specify ASCII or BINARY transfer mode.
- Download files (with wildcard support).
- Upload files (with wildcard support).
- Delete files.
- Rename files.
- Append files.
- Create, delete and rename directories.
- Asynchronous and synchronous file transfer.
- Real-time upload/download data transfer rate.
- Real-time number bytes received/sent for async transfers
- Create and remove server directories.
- Support for PROXY servers.
- Supports active and passive mode (use with firewalls) file transfers.
- Supports multiple concurrent FTP sessions.
- Resume (restart) file uploads and downloads from any offset.
- Change files names while being uploaded or downloaded.
- Can parse long directory listings.
- Can specify the FTP or data port.
- Can set minimum and maximum response waits.
- Specify the allowable data port range.
- All operations can be aborted.
- Supports S/KEY password encryption.
- Use on Internet or your own intranet (LAN).
- Is native Windows code but can be called from managed code.
- Will run on machines without .NET installed
- Works with all versions of Microsoft Visual FoxPro (VFP 3.0 through VFP 9.0)
- Can be used with Microsoft Visual Studio .NET
- Does **not** depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any program (in any language) capable of calling Windows API functions such as Visual C++, .NET, Visual Basic, Delphi, Xbase++, dBASE, COBOL, Access and Excel.
- Supports Windows XP through Windows 10.
- License covers all programming languages.
- Royalty free distribution with your compiled application.
- Can be purchased with or without source code [Ansi C].
- Updates are free for one year (Updates to source code are separate).
- Unlimited one-year email and phone support.
- Evaluation version is fully functional.

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the first manual (FCE_4FP.PDF) in the set.

- FCE4FP Programmer's Manual (FCE_4FP.PDF)
- FCE User's Manual (FCE_USR.PDF)
- FCE Reference Manual (FCE_REF.PDF)

The FCE Programmer's Manual (FCE_4FP.PDF) is the language specific (Visual FoxPro) manual dealing with compiler and programming issues such as installation and example programs. Read this manual first.

The FCE User's Manual (FCE_USR.PDF) discusses basic FTP processing as well as language independent programming issues such as application notes and includes purchasing and license information. Read this manual second.

The FCE Reference Manual (FCE_REF.PDF) contains details on each individual FCE function.

All manuals can be viewed online at http://www.marshallsoft.com/fce4fp.htm

1.3 Example Program

The following 32-bit FoxPro example demonstrates the use of some of the **FTP Client Engine** functions:

```
#INCLUDE FCE32CON.FOX
#INCLUDE KEYCODE.FOX

DECLARE INTEGER fceAttach in FCE32.DLL INTEGER Chan, LONG KeyCode
DECLARE INTEGER fceClose in FCE32.DLL INTEGER Chan
DECLARE INTEGER fceConnect in FCE32.DLL INTEGER Chan, STRING @Server, STRING @User,
                                       STRING @Pass
DECLARE INTEGER fceErrorText in FCE32.DLL INTEGER Chan, INTEGER Code,
STRING @Buffer, INTEGER BufLen
DECLARE INTEGER fceGetFile in FCE32.DLL INTEGER Chan, STRING @FileName
DECLARE INTEGER fceRelease in FCE32.DLL
DECLARE INTEGER fceSetMode in FCE32.DLL INTEGER Chan, INTEGER Mode
DECLARE INTEGER fceSetServerDir in FCE32.DLL INTEGER Chan, STRING @DirName

FTPserver = "ftp.marshallsoft.com" + Chr(0)
FTPuser = "anonymous" + Chr(0)
FTPpass = "you@your_isp.com" + Chr(0)
FTPdirectory = "pub/oem" + Chr(0)
FTPfilename = "fce-new.txt" + Chr(0)
* attach FCE
Code = fceAttach(1, FCE_KEY_CODE)
if Code < 0
  ? "Cannot attach FCE ", Code
  return
endif
* connect to FTP server
? "Connecting to ", FTPserver
Code = fceConnect(0, @FTPserver, @FTPuser, @FTPpass)
if Code < 0
  DisplayError(Code)
  Code = fceRelease()
  return
endif
? "Connected. Now downloading file..."
* change to proper directory
Code = fceSetServerDir(0, @FTPdirectory)
* set to ASCII xfer mode
Code = fceSetMode(0, ASC("A"))
* download the file
Code = fceGetFile(0, @FTPfilename)
if Code <= 0
   DisplayError(Code)
else
   ? "File downloaded."
endif
* log off
Code = fceClose(0)
Code = fceRelease()
? "Logged off."
return

Procedure DisplayError(ErrCode)
TempBuffer = SPACE(128)
Code = fceErrorText(0,ErrCode,@TempBuffer,128)
? Left(TempBuffer,Code)
return
```

In the example program above, **fceConnect** is called to connect to the FTP server as user "anonymous" and password "you@your_isp.com". The server directory is changed to "pub/other", the transfer mode is set to ASCII, and the file "products.txt" is downloaded. Lastly, the connection to the FTP server is closed and FCE is released.

Refer to the FCE Reference Manual (FCE_REF) for individual function details. Access online at http://www.marshallsoft.com/fce_ref.pdf

1.4 Installation

(1) Before installation of FCE4FP, a Visual FoxPro compiler should already be installed on your system and tested.

(2) FCE4FP34.ZIP (evaluation version) or FCExxxxx.ZIP (purchased version; xxxxx is the Customer ID).

(3) Run the installation program SETUP.EXE which will install all FCE4FP files, including copying FCE32.DLL to the Windows directory, which is normally C:\WINDOWS

(4) FCE4FP is ready to run! Compile and run FCEVER.PRG

1.5 Uninstalling

Uninstalling FCE4FP is very easy.

First, run UNINSTAL.BAT, which will delete FCE32.DLL from your Windows directory, typically C:\WINDOWS for Windows 98/ XP/2003-2012/Vista/Win7/Win8/Win10 or C:\WINNT for Windows NT/2000.

Second, delete the FCE project directory created when FCE4FP was installed.

1.6 Pricing

A developer's license for FCE4FP can be purchased for $115 ($295 with ANSI C source code to the DLLs). Purchasing details can be found in Section 1.3, "How to Purchase", of the FCE User's Manual (FCE_USR).

1.7 Updates

When a developer license is purchased, the developer will be sent a registered DLL plus a license file (FCExxxx.LIC, where xxxx is the Customer ID). The license file can be used to update the registered DLL for a period of one year from purchase. Updates can be downloaded from

http://www.marshallsoft.com/oem.htm

After one year, the developer license must be updated to be able to download updates. The license can be updated for $30 if ordered within one year of the original purchase (or previous update). Between one year and three years, licenses can be updated for $55. After three years, updates are $75.

Note that the registered DLL never expires.

Updates to the source code can be purchased for $100 in addition to the cost of the update.

Also see file UPDATES.TXT.

2 Library Overview

The **FTP Client Engine Library for Visual FoxPro** has been tested on multiple computers running Windows 98/ 2003-2012/XP/Vista/Win7/Win8/x64 and Windows NT/2000.

The FCE4FP library works with all versions of 32-bit Visual FoxPro.

The SETUP installation program will copy the FCE DLLs to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the FCE4FP files are copied to the directory specified (default \FCE4FP). Three sub-directories are created, as follows:

```
DOCS – All documentation files
APPS – All example code
DLLS – All DLL's
```

2.1 Dynamic Link Libraries

The FCE4FP library component is implemented using a Win32 dynamic link library (DLL). A DLL is characterized by the fact that it need not be loaded until required by an application program and that only one copy of the DLL is necessary regardless of the number of application programs that use it. Contrast this to the traditional static library that is bound to each and every application that uses it at link time.

An important advantage that DLLs have over other "popular" library formats such as VBX or OCX is that DLLs are callable by all Windows applications. Since DLLs are the building blocks of the Windows Operating System, they will not be replaced by a "newer technology".

2.2 Keycode

FCE32.DLL has a keycode encoded within it. The keycode is a 9 or 10-digit decimal number (unless it is zero), and will be found in the file KEYCODE.FOX. The keycode for the evaluation version is 0. The developer will receive a new key code after purchasing a license. SETUP copies the keycode to the /APPS sub-directory (folder). The KEYCODE is passed to **fceAttach**.

The keycode is not the customer ID (which is a 4 or 5 digit number).

If the error message (value -74) is returned when calling **fceAttach**, it means that the keycode in the application does not match the keycode in the DLL. After purchasing, it is best to remove the evaluation version of the FCE DLL from the Windows search path.

2.3 INCLUDE Files

All example programs include two files: `KEYCODE.FOX` and `FCE32CON.FOX`. The file `FCE32CON.FOX` contains all the necessary constants for FCE4FP, while the file `KEYCODE.FOX` contains the key code, as discussed in Section 2.2.

Since function declarations can not be in an `INCLUDE` file (at least through VFP version 9.0), they are listed in each program following the two `INCLUDE` files. The complete list of function declarations is also in the file `FCE32FUN.FOX`.

Due to the behavior of Visual FoxPro regarding `INCLUDE` files, it is strongly recommended that the `INCLUDE` files `KEYCODE.FOX` and `FCE32CON.FOX` be replaced with their contents in application programs (i.e., copy and paste contents) of the INCLUDE file.

2.4 Win32 STDCALL and DECLSPEC

FCE32 is written in ANSI C and is compiled using the _stdcall and _declspec keywords. This means that FCE4FP uses the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are neither leading underscores nor trailing "@size" strings added to function names.

Any Windows application program capable of calling the Windows API provided the proper declaration file is used may call the FCE32.DLL functions.

2.5 Dynamic Strings

The Visual FoxPro language uses a technique known as "garbage collection" to manage string space at runtime, and may be called internally at any time by the FoxPro runtime, asynchronous to what you may be doing in your code.

When passing a string buffer to a DLL function into which text will be copied, it is strongly recommended that the local string be allocated immediately before use. For example:

```
Code = fceConnect(0,@Server,@User,@Password)
if Code < 0
  * allocate buffer just before call
  Temp = SPACE(128)
  * put text in Temp
  Code = fceErrorText(1,Code,@Temp,128)
  ? Left(Temp,Code)
endif
```

This technique is not necessary for passing a string to a DLL function, only when passing a buffer to a DLL into which data is to be placed by the DLL function.

2.6 Null Terminated Strings

All strings returned from FCE functions are null terminated which means the end of the string is delimited by a Chr(0) character. These strings may be converted for FoxPro in one of two ways: (1) if the length of the string is known, use the FoxPro `LEFT` function: For example,

```
* get server IP address
TempBuffer = SPACE(TEMP_SIZE)
Code = fceGetString(0, FCE_GET_SERVER_IP, @TempBuffer, TEMP_SIZE)
if Code > 0
  ? "Server IP ", LEFT(TempBuffer, Code)
endif
```

If the length of the null terminated string is not known, use the FoxPro `AT` function to find the position of Chr(0). For example,

```
Offset = fceMatchFile(@ListBuffer,Offset,@NameBuffer,128,@FTPpattern,1)

P = AT(Chr(0), NameBuffer)
? "Downloading file " + Left(NameBuffer, P-1)
```

2.7 FoxPro Forms

FCE functions can be called from any Visual FoxPro code module, such as programs, classes, and forms. See the HELLO.SCT example form.

2.8 FTP Parameters

There are two types of FTP connections: private and anonymous. However, some FTP servers do not accept anonymous connections.

Three parameters are necessary in order to connect to an FTP server, as follows:

- Host name (or IP address) of the FTP server.
- User name.
- User password.

For private connections, the users account name and password must be specified.

For anonymous connections, the user name is "anonymous" and the password is the user's email address.

These FTP parameters are hard coded in most of the examples. However, these parameters could be read from the keyboard, from a file, from a dialog box at runtime, etc., as well as being hard coded.

Refer to the FCE User's Manual (FCE_USR) for more information regarding FTP protocol parameters.

2.9 Adding FCE to a VFP Program

- Add the FCE constants (found in FCE32CON.FOX) that will be used in the developer's application.
- Add the FCE function declarations (found in FCE32FUN.FOX) that will be called from the developer's application.

Refer to the example programs.

2.10 16-bit FoxPro

FCE4FP no longer supports Win16.

2.11 64-bit FoxPro

FCE4FP will support 64-bit (Win64) FoxPro if it is released by Microsoft. Note that Microsoft Visual Studio and Delphi XE2 support Win64 as does FCE.

2.12 Error Display

The error message text associated with FCE error codes can be displayed by calling **fceErrorText**. Each sample program contains examples of error processing.

Also see the file fceErrors.txt for a list of all FCE error codes.

3 Compiling Issues

FCE4FP works with all versions of 32-bit Visual FoxPro.

3.1 Compiling Programs

The example programs are compiled from the Visual FoxPro development environment. Before running the example programs, edit each program with your FTP parameters (see Section 2.8), as shown in the example program in Section 1.3 above. Server names can be IP addresses (in decimal dot notation) or the host name.

3.2 Compiling to an Executable

FoxPro programs end in ".PRG". They can be compiled to an executable using the FoxPro BUILD command.

For example, to create FCEVER.EXE from FCEVER.PRG in the C:\TEMP directory, type the following in the FoxPro command window:

```
BUILD PROJECT C:\TEMP\FCEVER FROM C:\TEMP\FCEVER
BUILD EXE C:\TEMP\FCEVER FROM C:\TEMP\FCEVER
```

FoxPro executables require VFP500.DLL and VFP5ENU.DLL (ENglish User), and may have to be copied from the VFP CDROM. If you are using an earlier or later version of FoxPro than version 5.0, substitute the appropriate DLL's for the above.

The FoxPro output display window will disappear as soon as your executable completes. In order to allow the user to control when the display window disappears, add the following code to your application, just before the final return.

```
?  " Type any key to exit..."
X = InKey(0)
```

3.3 Compiling FCE Source

The source code for the FCE DLL's is written in standard ANSI C (FCE32.C), and has been compiled using Microsoft Visual C++. The Win32 version is compiled with the STDCALL and DECLSPEC compiler keywords. Source code for the FCE library can be purchased at the same time as a FCE developer license is purchased.

FCE may also be compiled using Borland C/C++ or Watcom C/C++ compilers. If you recompile FCE32.C is compiled using Borland or Watcom compilers, the resulting DLL can only be used by applications compiled with the same compiler, unless the STDCALL and DECLSPEC keywords are specified.

For more information on the C/C++ version of FCE, download the latest version of FCE4C from our web site at http://www.marshallsoft.com/ftp-client-library.htm

4 Example Programs

All example programs are written for 32-bit FoxPro. Each has been tested and shows how to correctly use FCE functions. It suggested that the developer compile and run the example programs before developing an application using FCE4FP.

Each example program (with the exception of FCEVER.PRG, GET.PRG, and WinFTP) must be edited with the appropriate FTP protocol parameters before compiling. Refer to the FCE User's Manual (FCE_USR) for details.

Because of the peculiarity of Visual FoxPro regarding `INCLUDE` files, it is highly recommended that the `INCLUDE` files `KEYCODE.FOX` and `FCE32CON.FOX` be replaced with their contents.

Refer to Section 3.1 above for information on compiling and linking the example programs.
FCE functions may also be called from Visual FoxPro projects.

4.1 FCEVER

This simple program (FCEVER.PRG) displays the FCE version number, build number, and registration string taken from FCE32.DLL. Its purpose is to display the current version of the FCE DLL as well as to verify that FCE32.DLL is being found and loaded by Windows. The FCEVER program does not connect to your LAN (or the Internet).

This is the first program that you should compile and run.

4.2 GET

GET is a FTP client that connects to our FTP server anonymously and downloads the file "fce-new.txt". After compiling, start GET from the command line.

4.3 LIST

The LIST example program (LIST.PRG) logs onto the specified FTP server and lists all files. Edit LIST.PRG with your FTP server name (or IP address), user name, and password before compiling.

4.4 FIELDS

The FIELDS example program (LIST.PRG) logs onto the specified FTP server and lists all files by field. Edit FIELDS.PRG with your FTP server name (or IP address), user name, and password before compiling.

4.5 FTP

The FTP example program is a command line driven FTP client. Edit FTP.PRG with your FTP server name (or IP address), user name, and password before compiling.

4.6 FTP2

The FTP2 example program is the same as the FTP.PRG example, except that it uses the FCE direct mode rather than FCE indirect mode. Refer to the Users Manual (FCE_USR) for more information on direct and indirect mode. Edit FTP.PRG with your FTP server name (or IP address), user name, and password before compiling.

4.7 PROXY

The PROXY example program connects to an FTP server through a proxy server using the "USR@SERVER" protocol. Edit PROXY.PRG with your FTP server name (or IP address), user name, and password before compiling.

Refer to the FCE User's Manual (FCE_USR) for a discussion of proxy servers and proxy protocols; Section 3.7 "Proxy Servers" and Section 3.8 "Proxy Protocols" of the FCE User's Manual (FCE_USR), (http://www.marshallsoft.com/fce_usr.htm#Section_3.7). The text file PROXY2.TXT in the \FCE4FP\DOCS sub-directory provides information.

4.8 MGET

The MGET example program downloads files according to a wildcard pattern (using '?' and '*' characters). Edit MGET.PRG with your FTP server name (or IP address), user name, and password before compiling.

4.9 MPUT

The MPUT example program uploads files according to a wildcard pattern (using '?' and '*' characters). Edit MPUT.PRG with your FTP server name (or IP address), user name, and password before compiling.

Note that virtually all FTP servers do not allow files to be uploaded in anonymous mode.

4.10 SPEED

The SPEED example program connects to the MarshallSoft FTP server at ftp://ftp.marshallsoft.com and downloads a test file from the MarshallSoft FTP server and displays the time required. Use this program to see how long it takes to download files from FTP servers.

4.11 HELLO

The HELLO example form connects to a FTP server and verifies that it is operational by sending a NOOP command to the server.

From the VFP menu (File/Open), open the form HELLO.SCX (with "File of Type: Form"). When the "Form" menu tab appears on the VFP menu bar, choose "Run Form". This form can also be opened from the VFP command window by typing "`modify form \fce4fp\apps\winftp.scx`".

4.12 WinFTP

The WinFTP example form functions as a generic FTP client able to list, upload, and download files.

From the VFP menu (File/Open), open the form WinFTP.SCX (with "File of Type: Form"). When the "Form" menu tab appears on the VFP menu bar, choose "Run Form". This form can also be opened from the VFP command window by typing "`modify form \fce4fp\apps\winftp.scx`".

4.13 TIME

The TIME example program requests the "File Modification Time" for a file. However, not all FTP servers support the MDTM command.

5 Revision History

The FTP Client Engine DLL (FCE32.DLL) is written in ANSI C. All language versions of FCE (C/C++, Delphi, Visual Basic, PowerBASIC, FoxPro, dBase, Xbase++, FORTRAN, and COBOL) use the same FCE32.DLL.

Version 2.0: May 24, 2000

- The initial release of the Visual FoxPro version of FCE.

Version 2.1: February 5, 2001.

- Increased buffer size from 64 to 128 bytes for LocalDir and ServerDir.
- WriteBufferSize default increased to 512.
- Password not used if specified password has zero length.
- fceSetLocalDir() verifies that local directory is writable.
- Added FCE_SET_APPEND_MODE.
- Allow 128 character filenames.
- Added FCE_RENAME_DELIMITER to fceSetInteger().
- Corrected fceGetStatus(Chan, FCE_CONNECT_STATUS)
- Added FCE_SET_CLIENT_OFFSET and FCE_SET_SERVER_OFFSET.
- Rename file being up/downloaded by specifying "oldname:newname" for file.
- Added FCE_GET_LOCAL_IP to fceGetString.

Version 2.2: October 10, 2001.

- Default write buffer size increased from 512 to 1024 (WIN32 only).
- Added fceMatchFile function (used in multi-file uploads and downloads).
- Specify "User" as Chr(0) [in fceConnect] to skip USER and PASS processing.
- Performance improvements.
- Added MGET and PROXY example programs.

Version 2.3: December 5, 2002.

- Added FCE_SET_BLOCKING_MODE to control blocking (default ON) while connecting.
- Size of command buffer in fceCommand increased from 64 to 128 bytes.
- "WARNING: 226/250 not seen" written to log file rather than returning error.
- Added FCE_GET_ERROR_LINE.
- fceCommand sends CRLF with command in one network write.
- Added fceFileLength function.
- fceExtract handles line # 0.
- Added FCE_GET_QUEUE_ZERO (returns # times fceQueueLoad returns 0).
- Fixed problem with long server offsets (replaced "REST %d" with "REST %ld") in 16-bit version.
- Changed to 32 channels & 128 data ports (random time bits no longer used).
- fceGetList returns error if receive buffer is too small.

Version 2.4: May 26, 2004.

- Added FCE_SET_CONNECT_WAIT_IN_SECS.
- Added FCE_SET_MAX_RESPONSE_WAIT_IN_SECS.
- Added FCE_SET_MAX_LINE_WAIT_IN_SECS.
- FCE_NOT_COMPLETED returned if code 226 (or 250) not returned by control socket (in 2 tries).
- Number data ports changed to 2048 per channel for 1 and 2 channels
- Number data ports changed to 512 data ports per channel for 3 to 8 channels
- Number data ports changed to 128 data ports per channel for 9 to 32 channels
- DataPort mask corrected to 0x7FFF
- Added fceGetLocalFList function to get list of files in local directory.
- Added fceGetLocalFSize function to get the size of a file in the local directory.
- Added new MPUT and MGET example programs (using wildcards).

Version 2.5: July 27, 2005.

- LocalDir always stored with backslash as last character.
- fceWriteSocket makes up to 12 attempts to write.
- Add FCE_AUTO_LOG_CLOSE and FCE_CLOSE_LOG_FILE
- Adjusted wait time-outs.
- Added FCE_NO_GREETING error.
- Improved operation of fceGetInteger(Chan, FCE_GET_CONNECT_STATUS).
- Added GET.PGM example program.
- Added fceByteToShort and fceShortToByte functions.

Version 2.6: February 16, 2007

- Maximum PUT buffer size increased from 8K to 16K (16384)
- Recoded sleep wait in fceWriteSocket for improved upload performance.
- Added internal memory allocation debugging.
- Added GET_FULL_RESPONSE
- Close control socket whenever fceConnect fails (fixes socket leak problem).
- Added FCE_SET_FIRST_DATA_PORT and FCE_SET_LAST_DATA_PORT to fceSetInteger.
- Maximum data port extended to 65535.
- Added FCE_HIDE_PASSWORD to fceSetInteger.
- Added S/KEY authentication
- Added fceGetTicks()
- Added FCE_STATUS_BEFORE_WRITE to fceSetInteger.

Version 2.7: July 17, 2008 (32-bit version only)

- Fixed problem with non-blocking mode when connecting.
- Added FCE_LOCAL_DIR_IS_CDROM to fceSetInteger, which allows the local directory to be a read-only device such as a CDROM.
- Added FCE_DISABLE_SKEY to fceSetInteger, which disables S/KEY processing.
- Added MDTM example program.

Version 3.0: September 25, 2009

- Added support for 64 bit Windows (FCE64.DLL)
- Added fceIsConnected function
- Added fceToInteger function

Version 3.1: July 12, 2011

- Log file is now time-stamped
- Added diagnostics to fceFileLength
- Function fceGetLocalFList no longer counts subdirectories
- Added fceGetFileSize function
- Added fceGetFileTime function

Version 3.2: June 1, 2012

- Fixed Bug: Open control socket not always closed.
- Fixed Bug: Open listen socket not always closed.
- Added function fcePutDirFiles (uploads all files in directory)
- Added function fceGetDirFiles (downloads all files in directory)

Version 3.3: May 12, 2014

- Fixed problem with connecting w/o blocking.
- Fixed problem with server name being corrupted in the FCE log file.
- Added function fceGetSubDirs()
- Added FCE_SET_DEBUG_LEVEL to fceSetInteger().

Version 3.4: November 17, 2015

- Added debug diagnostics to fceSocketStatus().
- Fixed problem with connecting w/o blocking.
- Fixed problem with server name being corrupted in the FCE log file.
- Replaced FCE_EOF with FCE_CANNOT_OPEN if FCE log file cannot be created.
- Automatically adjust sleep times for slow FTP servers.

Check **http://www.marshallsoft.com/fce4fp.htm** for the latest version of our FTP client library software.