

# **FTP Client Engine**

## **Users Manual**

**(FCE\_USR)**

**Version 3.1**

**July 6, 2011**

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

Copyright (C) 2011  
All rights reserved

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

Voice : 1.256.881.4630  
email : [info@marshallsoft.com](mailto:info@marshallsoft.com)  
web : [www.marshallsoft.com](http://www.marshallsoft.com)

**MARSHALLSOFT** is a registered trademark of MarshallSoft Computing.

# TABLE OF CONTENTS

1	Introduction	Page 3
	1.1 Documentation	Page 3
	1.2 Technical Support	Page 5
	1.3 How to Purchase	Page 6
	1.4 Updates	Page 7
	1.5 Customer ID	Page 8
	1.6 License File	Page 8
2	FTP Client Library Overview	Page 9
	2.1 Keycode	Page 9
	2.2 Dynamic Link Library	Page 9
	2.3 GUI and Console Mode	Page 9
	2.4 Getting Started Using the Library	Page 9
3	Using FTP	Page 10
	3.1 FTP Basics	Page 10
	3.2 Private and Anonymous Access	Page 10
	3.3 ASCII and Binary Modes	Page 10
	3.4 Passive Mode	Page 10
	3.5 Socket Address In Use Error	Page 11
	3.6 Renaming Files on the Server	Page 11
	3.7 Proxy Servers	Page 11
	3.8 Proxy Protocols	Page 12
	3.9 Firewalls	Page 13
	3.10 Renaming Files "On The Fly"	Page 14
	3.11 Using Append Mode for Uploads	Page 14
	3.12 Using Append Mode for Downloads	Page 15
	3.13 Getting File Lengths	Page 15
	3.14 Adjusting Performance	Page 16
	3.15 Auto Dial	Page 17
	3.16 Secure FTP	Page 18
	3.17 FTP Passwords	Page 19
	3.18 S/Key Password Encryption	Page 19
	3.19 Progress Bars	Page 20
	3.20 Network Connectivity	Page 20
4	Theory of Operation	Page 21
	4.1 Indirect Method	Page 21
	4.2 Direct Method	Page 21
5	Development Languages Supported	Page 22
	5.1 Using FCE with Supported Languages	Page 22
	5.2 Using FCE with Unsupported Languages	Page 22
6	Resolving Problems	Page 23
7	Versions of FCE	Page 24
	7.1 Evaluation Version	Page 24
	7.2 Academic Version	Page 24
	7.3 Professional Version	Page 24
8	Legal Issues	Page 25
	8.1 License	Page 25
	8.2 Warranty	Page 25
9	FCE Function Summary	Page 26
10	FCE Error Return Code List	Page 27

## 1 Introduction

The **FTP Client Engine (FCE)** is a component DLL library providing direct control of the FTP protocol. The FCE component library can be used for both anonymous and private FTP sessions and can be used with any application capable of calling the Windows API.

A simple interface provides the capability to quickly develop FTP software applications to connect to any FTP server, navigate its directory structure, list files, upload files, delete files, append files, and download files using the FTP protocol.

FTP functions can easily be called from any program written in any programming language (such as C/C++, Visual Studio .NET, Visual C#, Delphi, Visual Basic, VB.NET, PowerBASIC, Visual FoxPro, dBase, Xbase++, COBOL, etc.) that is capable of calling Windows API functions.

The User's Manual applies to the **FTP Client Engine (FCE)** for all supported programming languages. It discusses FTP processing as well as language independent programming issues and provides purchasing and licensing information.

We have versions of the **FTP Client Engine SDK** for C/C++ (FCE4C), Delphi (FCE4D), Visual Basic (FCE4VB), PowerBASIC (FCE4PB), Visual FoxPro (FCE4FP), dBase (FCE4DB) and Alaska Xbase++ (FCE4XB). Purchase a developer license for one software development language and use it with all others. All versions of FCE use the same DLLs (FCE32.DLL and FCE64.DLL), however, the examples provided for each version are written and tested for the specified computer development language.

The **FTP Client Engine** DLLs (FCE32.DLL and FCE64.DLL) run under all versions of Windows (Windows 95, Windows 98, Windows ME, Windows 2000, Windows NT, Windows Server 2003, Windows XP, Windows Vista, and Windows 7. The 64-bit version (FCE64.DLL) runs with 64-bit applications running under Vista/x64 and Windows 7 /x64.

Fully functional versions of our **FTP Client** software components are provided so that the developer can test the **FCE** library in their environment. The evaluation version as well as a list of the many FTP Client library features provided can be found on our website at:

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

## 1.1 Documentation Set

The complete set of documentation consists of three manuals. This is the second manual (FCE\_USR) in the set.

- FCE4x Programmer's Manual (FCE\_4x.PDF)
- FCE User's Manual (FCE\_USR.PDF)
- FCE Reference Manual (FCE\_REF.PDF)

The FCE4x Programmer's Manual is the computer language specific manual. All language dependent programming issues including installation, compiling and example programs are discussed in this manual. Language specific manuals are as follows:

- FCE\_4C FCE Programmer's Manual for C/C++
- FCE\_4D FCE Programmer's Manual for Delphi
- FCE\_4VB FCE Programmer's Manual for Visual Basic
- FCE\_4PB FCE Programmer's Manual for PowerBASIC
- FCE\_4FP FCE Programmer's Manual for Visual FoxPro
- FCE\_4DB FCE Programmer's Manual for Visual dBase
- FCE\_4XB FCE Programmer's Manual for Xbase++

The FCE User's Manual (FCE\_USR) discusses FTP processing as well as language independent programming issues. License and purchase information is also provided. Read this manual after reading the FCE\_4x Programmer's Manual.

The FCE Reference Manual (FCE\_REF) contains details on each individual FCE function.

All documentation can also be accessed online at <http://www.marshallsoft.com/ftp-client-library.htm>

## 1.2 Technical Support

We want you to be successful in developing your applications using our FTP Client Library! We are committed to providing the best, robust component library that we can. If you have any suggestions or comments, please let us know.

If you are having a problem using FCE, refer to Section 6.0 “Resolving Problems”. If you still cannot resolve your problem, email us at

[support@marshallsoft.com](mailto:support@marshallsoft.com)

To avoid having your email deleted by our Spam scanners, begin the subject with “MSC HELP” or with the product name (FCE4C, FCE4VB, etc.). Zip up any attachments and send plain ASCII text email only.

Contact us by phone at 1.256.881.4630 between 7:00 AM - 7:00 PM CST Monday-Thursday and 7:00 AM -5:00 PM Friday.

The latest versions of our products are available on our web site at

<http://www.marshallsoft.com>

and on our anonymous FTP site at

<ftp://ftp.marshallsoft.com/pub>

Registered users can update (for a period of one year) to the latest DLL's at

<http://www.marshallsoft.com/update.htm>

### 1.3 How to Purchase

A developer license for the **FTP Client Engine Library** may be purchased for \$115 (USD) for electronic (email) delivery, or \$295 (USD) with ANSI C source code for the DLLs. This price is good for one year from the release date.

The fastest and easiest way to order is on our web site at

<http://www.marshallsoft.com/order.htm>

Multiple copy discounts (3 or more) and site licenses are available. Please call for details.

We accept American Express, VISA, MasterCard, Discover, checks in US dollars drawn on a US bank, International Postal Money Orders (such as Western Union), and purchase orders (POs) within the USA from recognized US schools and companies listed in Dun & Bradstreet.

You can also order by completing INVOICE.TXT (pro forma invoice) and emailing (info@marshallsoft.com), mailing (see our address at top), or faxing it to us. Our fax number will be provided upon request.

For credit card orders, be sure to include the account number, the expiration date, the exact name on the card, and the complete card billing address (the address to which the credit card bill is mailed- not the banks). Please include the Card Verification Code (last 3 numbers printed on the back of Visa, MasterCard and Discover cards, or the 4 numbers of the front of American Express cards.) The cardholder's signature is required on faxed orders.

The purchased package includes:

- FCE32 and FCE64 (for 64-bit programming environment) Library without the “evaluation info” screen.
- Free downloadable updates to the registered DLLs for one year.
- Free telephone and email support for one year.

### 1.3.1 Academic Discount

We offer an "academic price" with a 40% discount for prepaid email orders to faculty and full time students currently enrolled in any accredited high school, college, or university. The software must be used for educational purposes. The academic discount does **not** apply to source code.

To qualify for the discount, your school must have a web site and you must have an email address at your school. When ordering, ask for the "academic discount", or enter "student at" (or "faculty at") and your schools web site address (URL) in the comments field of the order form on our web site order page. Your order will be sent to your email address at your school.

This offer is not retroactive and cannot be used with any other discount. Products bought with academic pricing cannot be used for any commercial purpose.

### 1.3.2 Source Code

Source code is available for the purpose of re-compiling FCE32.DLL. Source code for the DLL library is standard ANSI C. The source code for FCE32.DLL is copyrighted by MarshallSoft Computing and may not be released in whole or in part.

There are two ways to order Source Code for the **FTP Client Engine Library SDK**.

- (1) Source Code can be ordered at the same time as the Developer's License for \$295 (for both).
- (2) Source Code can be ordered within one year of purchasing a Developer's License for \$200. After one year, a Developer's License update must be purchased prior to purchasing the source code.

## 1.4 Updates

When a developer license is purchased for the **FTP Client Engine Library SDK**, the developer will be sent a new set of DLLs plus a license file (FCExxxx.LIC) that can be used to update the registered DLL (does not include source code) for a period of one year from purchase. Updates can be downloaded from

<http://www.marshallsoft.com/update.htm>

After one year, the developer license must be updated to be able to download updates. The developer license can be updated for \$30 if ordered within one year from the original purchase (or previous update). After one year, licenses can be updated for \$55 (\$75 after 3 years).

Source code previously purchased may be updated for \$100 in addition to the cost of the update (\$30, \$55 or \$75).

Note that the registered DLL's never expire.

## **1.5 Customer ID**

The Customer ID is the 4 or 5 digits following the product name (FCE) in the license file. For example, customer 12345 would receive license file **FCE12345.LIC**. Provide the Customer ID in the SUBJECT of an email when contacting us for technical support (FCE4C 12345).

## **1.6 License File**

A license file, FCExxxx.LIC, where “xxxx” is the 4 or 5- digit customer ID is provided with each developer license. The license file is an encrypted binary file used for updating FCE as explained in section 1.4 “Updates”. The license file is required in order to create (or update) the registered DLLs. The license file can be found in the /DLLS directory created after SETUP is run.

## 2 FTP Client Library Overview

The **FTP Client Engine Library** has been tested on multiple computers running Windows 95/98/Me/2003/XP/Vista/Vista x64/Win 7/Win 7 x64 and Windows NT/2000.

### 2.1 Keycode

When a developer license is purchased, the developer will receive a new set of DLLs and a keycode for the FCE DLL's. Pass this keycode as the argument to **fceAttach**. The keycode will be found in the file named "KEYCODE". The keycode for the evaluation version is 0. The keycode for the registered version will be a unique 9 or 10 digit number. Note: Your keycode is NOT your Customer ID/Registration number.

### 2.2 Dynamic Link Library

The **FTP Client Engine Library SDK** includes a Win32 [FCE32.DLL] and Win64 [FCE64.DLL] dynamic link libraries (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 a static library that is bound at link time to each and every application that uses it.

### 2.3 GUI and Console Mode

FCE functions can be called from WIN32 console mode programs as well as GUI programs. A "console mode" program is a Windows 95/98/Me/NT/2000/2003/XP/Vista WIN32 command line program running in a command window. Although console mode programs look like DOS programs, they are WIN32 programs that have access to the entire Windows address space.

### 2.4 Getting Started Using the Library

The first **FTP Client Engine (FCE)** function that should be called is **fceAttach**, which initializes the FCE library and allocates necessary resources. **fceAttach** is typically called in the initialization section of your application.

After **fceAttach** is called, you are ready to connect to a FTP server with **fceConnect**. Once connected, you are ready to call the other FCE functions.

After completing your FTP session, the connection to the server can be closed with **fceClose**. The **fceClose** function should not be called if the previous **fceConnect** failed.

Before exiting your application, **fceRelease** should be called. **fceRelease** should not be called if **fceAttach** failed.

The best way to get familiar with FCE is to try out one of the example programs. The example programs are described in the FCE4x Programmer's Manual. . The "x" in FCE\_4x specifies the host programming language such as C for C/C++, VB for Visual Basic, etc. the example source is written in.

- FCE\_4C FCE Programmer's Manual for C/C++
- FCE\_4D FCE Programmer's Manual for Delphi
- FCE\_4VB FCE Programmer's Manual for Visual Basic
- FCE\_4PB FCE Programmer's Manual for PowerBASIC
- FCE\_4FP FCE Programmer's Manual for Visual FoxPro
- FCE\_4DB FCE Programmer's Manual for Visual dBase
- FCE\_4XB FCE Programmer's Manual for Xbase++

## 3 Using FTP

### 3.1 FTP Basics

The FTP (File Transfer Protocol) protocol is defined by Internet document RFC 959. It is used to copy files between a FTP client and a FTP server over a TCP/IP connection using well-known port 21.

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 a FTP server, as follows:

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

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.

### 3.2 Private and Anonymous Access

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

Some FTP servers allow "anonymous" access, which is usually download only. For anonymous connections, the user name is "anonymous" and the password is the user's email address.

### 3.3 ASCII and Binary Access Modes

The default FTP access mode is ASCII text. In order to upload or download any file that is not ASCII text, the transfer mode must be set to binary with the **fceSetMode** function first. A binary file uploaded or downloaded in ASCII mode may be corrupted.

### 3.4 Passive Mode

FCE supports "Passive" mode. Passive mode means that the server specifies the data port rather than the client when listing or transferring files. Using passive mode is often necessary to get past a firewall.

Passive mode is an optional FTP command, so that although most FTP servers support passive mode, there are some that do not. Passive mode can be enabled by calling:

```
fceSetInteger(Chan, FCE_SET_PASSIVE, 1)
```

The use of passive mode is recommended when possible.

### 3.5 "Socket Address Already In Use" Error.

The FTP protocol specifies that data sockets are reserved after use for some fixed period of time. This means that a data socket cannot be immediately re-used. When making multiple calls to list or transfer files, FCE will increment the data socket number. However, if you terminate and then restart your application while the FTP server still has your last data socket reserved, and attempt to list or transfer files, you will get the error "socket address already in use". There are three solutions to this problem:

- (1) Wait a minute or so, then try again.
- (2) Use PASSIVE mode [`fceSetInteger(Chan, FCE_SET_PASSIVE, 1)`].
- (3) Specify the port yourself by calling `fceSetInteger(Chan, FCE_SET_DATA_PORT, port-number)`.

### 3.6 Renaming Files on the Server.

Files can be renamed on the FTP server by using the `fceCommand` function. For example, to rename the file "oldname.txt" to "newname.txt" on the server, use (C/C++ example):

```
// rename "oldname.txt" to "newname.txt" on the server
Code = fceCommand(Chan, "RNFR oldname.txt")
Code = fceCommand(Chan, "RNT0 newname.txt")
```

Also refer to Section 3.10, 'Renaming Files "On The Fly"'.

### 3.7 Proxy Servers

In order to use a FTP proxy server, the FTP client connects to the proxy server, and then the proxy server connects to the remote FTP server. The sequence of connection events is as follows:

- The client connects to the FTP proxy server using the FTP port as required by the proxy server.
- The client provides authentication information to the proxy server, if required.
- The client provides user name and password for the remote FTP server to the proxy server.
- The FTP proxy server connects to the remote FTP server.

The proxy server requires that a particular port (not well known port 21) be used when connecting to it. The proxy server port number is specific to the particular proxy server. Refer to the documentation for the specific FTP proxy server to find out the port number and the method for specifying connection parameters such as the user name and password.

The most common method for providing remote server parameters to a proxy server is sometimes known as "PROXY USER". For example, to connect to ftp.marshallsoft.com as user "anonymous" with password "test@marshallsoft.com" through the proxy server running at 10.0.0.1 on proxy port 4421;

```
fceSetInteger(0, FCE_SET_FTP_PORT, 4421)
fceConnect(0, "10.0.0.1", NullString, NullString)
fceCommand(0, "USER anonymous@ftp.marshallsoft.com")
fceCommand(0, "PASS test@marshallsoft.com")
```

Note that the user name and password in the above example are for the remote FTP server, not for the proxy FTP server. The "NullString" refers to a string (or buffer) in which the first byte is a hex zero. Refer to the PROXY example for a complete program.

Check the documentation provided with a particular proxy server for a description of the proxy protocol that it requires. Also refer to Section 3.8, "Proxy Protocols".

## 3.8 Proxy Protocols

There are no formally defined proxy connection protocols. However, the following lists the most common proxy connection protocols. Refer to the PROXY example program for an example using the "PROXY USER" connection protocol. Note that ProxyUser and ProxyPass are not always used. The following parameters are used for connection to a remote FTP site through a proxy server.

```
ProxyServer = Proxy server name or IP address.  
ProxyUser   = Proxy user name (optional).  
ProxyPass   = Proxy password (optional).  
ProxyPort   = Proxy port number. Consult your proxy documentation.  
  
RemoteServer = Remote FTP server name or IP address.  
RemoteUser   = Remote FTP server user name.  
RemotePass   = Remote FTP server password.
```

### 3.8.1 "RemoteUser@RemoteServer" Protocol (PROXY USER)

```
USER RemoteUser@RemoteServer  
PASS RemotePass
```

### 3.8.2 "User After Login" Protocol. Also known as the "Double Login" protocol.

```
USER ProxyUser  
PASS ProxyPass  
USER RemoteUser@RemoteServer  
PASS RemotePass
```

### 3.8.3 "ProxyUser@RemoteServer ProxyUser" Protocol.

```
USER ProxyUser@RemoteServer ProxyUser  
PASS RemotePass
```

### 3.8.4 "ProxyUser@RemoteUser@RemoteServer" Protocol.

```
USER ProxyUser@RemoteUser@RemoteServer  
PASS RemotePass
```

### 3.8.5 "Proxy Open" Protocol.

```
OPEN RemoteServer  
USER RemoteUser  
PASS RemotePass
```

Check the documentation provided with a particular proxy server for a description of the proxy protocol that it requires.

### 3.9 Firewalls

Firewalls operate transparently with respect to all TCP/IP programs, monitoring inbound and outbound traffic. Firewalls can filter packets based on their contents, source addresses, destination addresses, and port numbers. If the traffic meets the criterion of the firewall, it is allowed, otherwise it is not.

Firewalls are often combined with proxy servers. Firewalls usually require the use of passive mode. Refer to Section 3.4 "Passive Mode".

When a client program attempts to connect to a FTP server, the first thing that the server does is accept the connection, then send its greeting message - all before the actual logon. If a server's greeting message is not received, then one of the following may have occurred: an incorrect server name or IP address was specified, the wrong port (a few servers don't use standard port 21) was specified, there is a connectivity problem (problem with cable, DSL, Windows itself, etc.), or a firewall is blocking the connection.

If a connection to the server is successful, and all commands work up to the point prior to attempting to get a listing or upload/download a file, then the problem is almost always that the data port chosen is being blocked on one side or the other. Often using passive mode (which allows the server to choose the data port) will resolve this problem, although passive mode is not supported by all servers.

If the connection is being blocked by a local firewall (as opposed to the server's firewall) then the local firewall will need to be reconfigured to unblock a range of data ports. Note that a range of data ports can be specified by using:

```
fceSetInteger(FCE_SET_FIRST_DATA_PORT, first-data-port-to-use)
fceSetInteger(FCE_SET_LAST_DATA_PORT, last-data-port-to-use)
```

Once a data port is used, the port is freed, but it cannot be reused for a period of time determined by the server. To be on the safe side, use one data port for each file listing/upload/download that will occur within any 60-second time period.

### 3.10 Renaming Files "On The Fly"

Files can be renamed as they are being uploaded or downloaded by specifying the filename in "oldname:newname" format. For example, to download the file "serverfile.txt" from an FTP server renamed as "myfile.txt" on the client computer, call

```
fceGetFile(Chan, "serverfile.txt:myfile.txt")
```

Note that there are no extra spaces in the filename string.

The default filename delimiter is the ':' character, but can be changed to any character necessary. For example, to change the filename delimiter from ':' to '\$', call

```
fceSetInteger(Chan, FCE_SET_RENAME_DELIMITER, '$')
```

### 3.11 Using Append Mode for Uploads

Provided that a FTP server supports it, uploads can be appended to the existing file of the same name (on the server). This can be done by setting append mode and (optionally) a file offset before calling `fcePutFile`.

For example, to append file "myfile.txt" to the server,

```
fceSetInteger(Chan, FCE_SET_APPEND_MODE, 1)
fcePutFile(Chan, "myfile.txt")
```

To append "myfile.txt", but begin reading the file at file offset 1024 on the client computer,

```
fceSetInteger(Chan, FCE_SET_APPEND_MODE, 1)
fceSetInteger(Chan, FCE_SET_CLIENT_OFFSET, 1024)
fcePutFile(Chan, "myfile.txt")
```

In order to continue an interrupted upload, the offset used should be the existing size of the file on the server. The client offset (if specified) applies only to the client computer not to the server. The server file offset cannot be specified for uploads. Append mode and the file offset must be set for every file before calling **fcePutFile**.

### 3.12 Using Append Mode for Downloads

Provided that an FTP server supports it, downloads can be appended to the existing file of the same name on the client computer. This is done by setting append mode, a server file offset, and a client file offset before calling **fceGetFile**.

For example, to download file "myfile.txt" beginning at file offset 1024 on the server to the client computer, also at file offset 1024,

```
fceSetInteger(Chan, FCE_SET_APPEND_MODE, 1)
fceSetInteger(Chan, FCE_SET_SERVER_OFFSET, 1024)
fceSetInteger(Chan, FCE_SET_CLIENT_OFFSET, 1024)
fceGetFile(Chan, "myfile.txt")
```

In order to continue an interrupted download, both the client file offset and the server file offset should be set to the existing size of the file on the client computer. Append mode and the file offsets must be set for every file before calling **fceGetFile**.

### 3.13 Getting File Lengths

If your FTP server supports the "extended" FTP command "SIZE", the `fecGetFileSize` command can be used to get the size of a file on the FTP server. Otherwise, the file size will have to be retrieved from the file listing.

According to the FTP protocol standard, there is no standard for the format of full FTP listings returned by the server. The format will vary by host operating system, and sometimes between various makes of FTP servers. However, most servers return the full listing exactly as reported by the host operating system. For UNIX, it is typically "ls -l". For Windows, it is typically "dir".

To get the file length of a particular file, request a full listing entry by calling

```
fceGetList(Chan, FCE_FULL_LIST, DataBuffer, DataBufferSize)
```

then call **fceExtract** to extract each field in turn. One of the fields, depending on the host's operating systems, will be the field length.

## 3.14 Adjusting Performance

### 3.14.1 Max Performance

A few FTP servers have trouble receiving large buffers sent by the client. However, for most servers, the write performance can be increased by enlarging the write buffer size from 1024 (default value) to 8192 and reducing the sleep time set to 0:

```
fceSetInteger(0, FCE_SET_WRITE_BUFSIZE, 8192)
fceSetInteger(0, FCE_SET_SLEEP_TIME, 0)
```

If uploading is much slower than you believe that it should be, try calling

```
fceSetInteger(0, FCE_STATUS_BEFORE_WRITE, 0)
```

This will direct FCE to not check the WRITE status of a socket before writing to it.

### 3.14.2 Slow Servers

There are some FTP servers that, for various reasons, cannot keep up with the load of FTP requests from FTP clients. In order to make it easier for those servers, FCE can be programmed to run slower. For example,

```
fceSetInteger(0, FCE_SET_SLEEP_TIME, 150)
fceSetInteger(0, FCE_SET_MIN_LINE_WAIT, 250)
fceSetInteger(0, FCE_SET_MIN_RESPONSE_WAIT, 3000)
```

## 3.15 Auto Dial

### 3.15.1 Auto Dial for Windows 95/98/Me

To allow Dial-Up Networking (DUN) to dial up an ISP when the accessing the Winsock (WIN32 only):

(1) Open the DUN folder in "My Computer", and choose "Connections/Settings" from menu bar. Uncheck "prompt for information before dialing" and choose "Don't prompt to use Dial-Up Networking".

(2) Use the Windows REGEDIT program to change value "00 00 00 00" to the value "00 00 00 01" in the Windows Registry for the entry

HKEY\_CURRENT\_USER/Software/Microsoft/Windows/CurrentVersion/InternetSettings/EnableAutodial

(3) Use the Windows REGEDIT program to change value "00 00 00 00" to the value "00 00 00 01" in the Windows Registry for the entry

HKEY\_CURRENT\_USER/Software/Microsoft/Windows/CurrentVersion/InternetSettings/EnableAutodisconnect.

(4) Use the Windows REGEDIT program to change value "14 00 00 00" to the value "01 00 00 00" in the Windows Registry for the entry

HKEY\_CURRENT\_USER/Software/Microsoft/Windows/CurrentVersion/InternetSettings/DisconnectIdleTime

This changes the idle time (until disconnect) from 20 minutes (hex 14) to one minute.

### 3.15.2 Auto Dial for Windows NT

Windows NT and XP users can control auto dialing by editing the setting in the "Dial-Up Networking" (DUN) window. Choose "More", "User Preferences", then "Appearance".

The Dial-Up Networking window can also be displayed by executing the Microsoft Windows program RASPHONE.EXE.

### 3.15.3 Auto Dial for Windows 2000/XP/Vista

Start Windows help, then type "autodial", then click "configuring". Follow all directions, including "notes". Note that the "Remote Access Auto Connection Manager" must be enabled.

### 3.15.4 MarshallSoft DUN Dialer

The **MarshallSoft DUN Dialer** (MDD) can dial up (and hang up) under program control. MDD works under all Windows 32-bit operating systems (Win 95, 98, Me, XP, Vista, NT, 2000,2003).

For more information, see the MDD product page at [www.marshallsoft.com/mdd4c.htm](http://www.marshallsoft.com/mdd4c.htm) (for C/C++), [www.marshallsoft.com/mdd4vb.htm](http://www.marshallsoft.com/mdd4vb.htm) (for Visual Basic), etc. The cost for MDD is discounted when ordered at the same time as FCE.

## **3.16 Secure FTP**

The FTP protocol itself is not secure. If sensitive or private data is to be exchanged, some form of FTP security should be considered.

Unfortunately, there are quite a few differing solutions to FTP security, and none of them are part of the standard FTP protocol. There are, however, several solutions to the problem of FTP security. The FTP Client Engine supports 3.16.1 through 3.16.4 below as well as S/KEY password encryption (section 3.18).

### **3.16.1 Use a secret port.**

Changing the FTP server's port provides a minimal degree of security. The idea is that hackers will not know which port to attack. The port chosen should be changed on a regular basis. Most FTP servers and clients can use non-standard ports.

### **3.16.2 Encrypt all data.**

Encrypt all files using strong encryption such as "Pretty Good Privacy". Although hackers may be able to intercept FTP passwords and download copies of your encrypted files, they will not be able to decode them unless they know the encryption key. Standard FTP clients and servers can be used.

### **3.16.3 Implement VPN.**

Set up a "Virtual Private Network", which will encrypt all traffic between VPN nodes. VNP products may be either software or hardware, and are widely available. They also allow the use of standard FTP servers and clients.

### **3.16.4 Use Secure FTP proxy servers.**

Similar to VPN, secure FTP proxy servers can be used to automatically encrypt all traffic between two end points. Using secure FTP proxy servers allows the use of standard FTP servers and clients.

### **3.16.5 Use SSL, TSL, SSH, or SFTP protocols.**

Use one of several "secure" FTP servers that typically implement SSL, TSL, or SFTP security protocols. This approach also requires the use of an FTP client using the same security protocol and requires X.509 certificates. Currently the FTP Client Engine does not support these protocols, however, they are included in the documentation for completeness.

### 3.17 FTP Passwords

When a connection is made to a FTP server, a user name and password are normally supplied. However, occasionally the FTP server will be configured to require "no password". This can mean two different things:

- (1) The PASS command should not be sent. To do this, pass an empty string (a string whose first element is a NUL) as the password argument to **fceConnect**.
- (2) The PASS command should be sent by itself. To do this, pass a string consisting of a single space character as the password argument to **fceConnect**.

### 3.18 S/KEY Password Encryption

The S/KEY "One-Time Password System", defined by RFC 2289, encrypts passwords before sending them. The actual password is never transmitted in the clear as is the case for standard FTP.

A FTP server that supports the one time password system will respond to the client's USER command with the challenge string "otp-md5 <sequence> <seed>" to which the client responds with the 6 word password phrase as calculated from the users password according to RFC 2289. For example,

```
R: 220 Serv-U FTP Server v6.3 for WinSock ready...
S: USER mike
R: 331 Response to otp-md5 999 hello required for skey.
S: PASS NEIL JULY NONE EMIT FLEW MUDD
R: 230 User logged in, proceed.
```

If "otp-md5" is not seen, FCE sends the user's password normally (in the clear) as required by the FTP protocol. Thus, S/KEY password encryption is performed automatically when FCE connects to an S/KEY enabled FTP server.

The function `fceGetInteger(0, FCE_SKEY_WAS_SEEN)` will return TRUE (non zero) if the challenge string "otp-md5" was seen in the server's response to the client's USER command.

There are several FTP servers that support S/KEY password encryption. For Windows,

- Titan FTP Server (<http://www.southernrivertech.com>)
- Serv-U FTP Server (<http://www.serv-u.com>)
- Globalscape FTP Server (<http://www.globalscape.com>)

### 3.19 Progress Bars

Direct mode (calling **fceDriver**) must be used in order to periodically (for example, each second) get the transfer byte count as the upload/download progresses.

In order to implement an upload progress bar, it is necessary to know the size of the file to be uploaded, which can be found by calling **fceGetLocalFSize**. For implementing a download progress bar, the size of the file on the server must be known. This is complicated by the fact that there is NO standard (long) listing format. However, the file size is almost always the first numeric field (beginning with the third), and can be found by calling **fceFileLength**.

The percentage uploaded is computed as

```
BytesWritten = fceGetInteger(0, FCE_GET_FILE_BYTES_SENT)
FileSize = fceGetLocalFSize(0, FilenamePointer)
PerCentage = (100 * BytesWritten) / FileSize
```

and the percentage downloaded is computed as

```
BytesRead = fceGetInteger(0, FCE_GET_FILE_BYTES_RCVD)
FileSize = fceFileLength(LongListingPointer, 3, 7)
PerCentage = (100 * BytesRead) / FileSize
```

Also see the WINFTP example program that displays the percentage uploaded/downloaded at runtime.

### 3.20 Network Connectivity

It is possible to detect loss of network connectivity in some situations by calling

```
fceIsConnected(Channel)
```

However, depending on many factors, loss of network connectivity cannot always be detected. The only 100% method of detecting loss of connectivity to the server is to send a "NOOP" command to the server and then reading its response. Calling the function **fceHello** can perform this.

```
fceHello(Channel)
```

## 4 Theory Of Operation

The **FTP Client Engine (FCE)** is state driven. This means that each call to FCE functions (that access the server) is broken down into sequential steps, each of which can be performed within a second or so. There are two ways in which FCE is used: (1) indirect use of the state engine, and (2) direct use of the state engine.

### 4.1 Indirect Method

The first (or "indirect") way to use the FCE library is to allow all FCE function calls to automatically call the FCE driver (**fceDriver**) before returning. This is the default way that FCE operates.

The major advantage of this approach is that each FCE function returns only after it has completely finished. The disadvantage of this approach is that some functions may run for a considerable amount of time during which time the calling application must wait.

Refer to the GET example program for an illustration of this approach.

### 4.2 Direct Method

The second (or "direct") way that the FCE state driver is used is to call it (**fceDriver**) directly. In order to operate this way, the function **fceSetInteger** must be called to set the AUTO\_CALL flag to off:

```
fceSetInteger(Chan, FCE_SET_AUTO_CALL_DRIVER, 0)
```

After the above statement is executed, the state driver (**fceDriver**) must be called after all of the other FCE functions that access the server. For example (code example),

```
... enable direct mode (disable indirect mode).
fceSetInteger(Chan, FCE_SET_AUTO_CALL_DRIVER, 0)
... connect to server.
Code = fceConnect(...)
If Code < 0 Then
  ... handle error here.
End If
... run the driver.
Loop
  ... call the driver
  Code = fceDriver(Chan)
  If Code < 0 Then
    ... handle error here.
    Exit Loop
  End If
  If Code = 0 Then
    ... fceDriver has finished.
    Exit Loop
  End If
  ... display progress or do other processing here.
End Loop
... enable indirect mode (disable direct mode).
fceSetInteger(Chan, FCE_SET_AUTO_CALL_DRIVER, 1)
```

The major advantage of the direct approach is that the calling application can perform other work such as reporting the progress of large downloads. The disadvantage is the extra code that must be written to call **fceDriver**.

Refer to the WINFTP example program for an illustration of this approach.

## 5 Development Languages Supported

We have versions of the **FTP Client Engine (FCE)** component library for C/C++ and .NET (FCE4C), Borland Delphi (FCE4D), Visual Basic and VB.NET (FCE4VB), PowerBASIC (FCE4PB), Visual FoxPro (FCE4FP), dBase (FCE4DB), and Alaska Xbase++ (FCE4XB). All versions of FCE use the same DLLs (FCE32.DLL and FCE64.DLL). Evaluation versions for these may be downloaded from our website at

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

The **FTP Client Engine** DLL's can also be used with any application written in any language capable of calling the Windows (95/98/Me, NT/2000/XP/2003/Vista) API.

### 5.1 Using FCE with Supported Languages.

Once a developer license is purchased for a particular programming language version of the **FTP Client Engine SDK (FCE)**, the same license can be used with all other supported programming languages. Supported languages are C/C++, Visual Basic, PowerBASIC, Delphi, Visual FoxPro, Visual dBase, Xbase++, and (Fujitsu) COBOL.

For example, assume that you have previously downloaded and installed the registered version of FCE4C and now you want to also call FCE functions from Visual Basic.

1. Make a backup copy of FCE32.DLL and FCE64.DLL found in the Windows directory (\WINDOWS or \WINNT).
2. Download and install the evaluation version of FCE4VB (<http://www.marshallsoft.com/fce4vb.htm>) into a separate directory from FCE4C.
3. Compile and run the Visual Basic FCEVER example program found in the APPS directory created in step 2 above. It should display the pop-up evaluation screen.
4. Restore FCE32.DLL and FCE64.DLL saved in step 1 above.
5. Paste the key code value found in (the registered version of FCE4C) KEYCODE.H into KEYCODE.BAS.
6. Run the Visual Basic FCEVER example program again. It should no longer display the pop-up screen.

A quicker and easier way would be to request multiple programming versions of FCE when a developer license is purchased. There is no additional charge.

### 5.2 Using FCE with Unsupported Languages

The **FTP Client Engine** DLL can be used with any application written in any language capable of calling the Windows (95/98/ME, NT/2000/XP/2003/Vista) API.

Declaration files have been defined for the following languages:

C/C++ and .NET	FCE.H
Visual Basic and VB NET	FCE32.BAS
VBA (Excel, Access,...)	FCE32.BAS
PowerBASIC	FCE32.PBI
Borland Delphi	FCE32.PAS
Fujitsu COBOL	FCE32.CBI
Fortran	FCE32ABS.INC and FCE32DEC.INC
Visual FoxPro	FCE32.FOX
Visual dBase	FCE32.CC
Alaska Xbase++	FCE32.CH

Additional declaration files will be added. Email us if you need a declaration not listed above. If you have interfaced FCE to an unusual language, email us the declaration file!

## 6 Resolving Problems

First, be sure you are passing the proper keycode. Refer to section 2.1 "Keycode". Before attempting to run any of the example programs, you should already be able to connect to the Internet (or your TCP/IP LAN) and run your normal FTP client, such as the Microsoft command line FTP client FTP.EXE (in the Windows directory).

If you have trouble connecting to an FTP server, try using the IP address instead of the server name. If using the IP address works but the server name does not, the problem lies with the Domain Name System (DNS) lookup. If this does not solve your connection problem, try connecting using TELNET, located in the Windows or Windows/System32 .

If you cannot get your application to run properly, first compile and run the example programs. If you call us to report a possible bug in the library, the first thing we will ask is if the example programs run correctly. All example programs have been compiled and tested.

Be sure to test the code returned from all FCE functions. Then, call **fceErrorText** to get the text associated with the error code. All functions return an integer code. Negative values are always errors.

For example (C Example):

```
Code = fceConnect(0,"ftp.marshallsoft.com","mike", "mike");
if(Code<0)
    {static char Buffer[64];
      fceErrorText(0,Code,Buffer,64);
      printf("Error %d: %s\n", Code, Buffer);
    }
```

Another good idea is to turn on logging by calling

```
fceSetString(Chan, FCE_LOG_FILE, logfilename)
```

If you encounter a problem that you cannot resolve, email us at [info@marshallsoft.com](mailto:info@marshallsoft.com). To avoid having your email deleted by our Spam scanners, begin the subject with FCE or MSC HELP. Zip up any attachments and send plain ASCII text email only.

If you still get the evaluation screen (popup window) after purchasing a developer license, the problem is that Windows is finding the evaluation DLL before the registered DLL. The solution is to delete (or zip up) all evaluation versions of FCE32.DLL (or FCE64.DLL) and run the SETUP program again.

If you get "error -74" when calling **fceAttach**, the problem is that the keycode passed to **fceAttach** does not match the keycode in the DLL's. This is caused by (1) using the evaluation keycode (value = 0) with the registered DLL, or (2) using the registered keycode with the evaluation DLL.

We recommend the following steps if you believe that you have discovered a bug in the library:

- (1) Create the smallest, simplest test program possible that demonstrates the problem.
- (2) Document your exact machine configuration and what error the test program demonstrates.
- (3) Email to [support@marshallsoft.com](mailto:support@marshallsoft.com) the example source and log file.

If the problem is an error in the library and can be solved with an easy work-around, we will publish the work-around. If the problem requires a modification to the library, we will make the change and make the modified library available to our customers without charge.

Review the FCE Programmer's and Reference Manuals. We also suggest reading Section 3 "Using FTP".

## **7 Versions of FCE**

The **FTP Client Engine (FCE)** component library is available in three versions. All three versions have identical functionality.

### **7.1 Evaluation Version**

The evaluation version can be differentiated from the other two versions by:

- (1) The evaluation reminder screen is displayed at startup.
- (2) The evaluation version may not be used for commercial purposes.
- (3) The evaluation version may be used for 60 days.

### **7.2 Academic Version**

The academic version can be differentiated from the other two versions by:

- (1) There is no evaluation reminder screen.
- (2) The academic version is for academic purposes only and may not be used for commercial purposes.

### **7.3 Professional Version**

The professional version can be differentiated from the other two versions by:

- (1) There is no evaluation reminder screen.
- (2) The DLL is branded with your company name.
- (3) Your branded DLLs may be distributed with your compiled applications.

The professional version compiled DLL may be distributed royalty free with your compiled applications as specified by the software license. However, the Keycode to the DLL can NOT be distributed. The Professional version may be used for commercial purposes. Licensing information is provided in Section 10.1

## **8 Legal Issues**

### **8.1 License**

This license agreement (LICENSE) is a legal agreement between you (either an individual or a single entity) and MarshallSoft Computing, Inc. for this software product (SOFTWARE). This agreement also governs any later releases or updates of the SOFTWARE. By installing and using the SOFTWARE, you agree to be bound by the terms of this LICENSE. If you do not agree to the terms of this LICENSE, do not install or use the SOFTWARE.

MarshallSoft Computing, Inc. grants a nonexclusive license to use the SOFTWARE to the original purchaser for the purposes of designing, testing or developing software applications. Copies may be made for back-up or archival purposes only. This product is licensed for use by only one developer at a time. All developers working on a project that includes a MarshallSoft Software SDK, even though not working directly with the MarshallSoft SDK, are required to purchase a license for that MarshallSoft product.

The academic registered DLL's may not be distributed under any circumstances, nor may they be used for any commercial purpose.

The professional registered DLL's may be redistributed (without royalty) as part of the user's compiled application. The registered DLL's may NOT be distributed as part of any software development system (compiler or interpreter) without our express written permission. When the software is registered, a key-code will be provided, which enables access to the registered DLL's. This key-code may NOT be distributed or made known.

The SOFTWARE is owned by MarshallSoft Computing, Inc. and is protected by United States copyright laws and international treating provisions. This SOFTWARE is being licensed and not sold.

### **8.2 Warranty**

MARSHALLSOFT COMPUTING, INC. DISCLAIMS ALL WARRANTIES RELATING TO THIS SOFTWARE, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND ALL SUCH WARRANTIES ARE EXPRESSLY AND SPECIFICALLY DISCLAIMED. NEITHER MARSHALLSOFT COMPUTING, INC. NOR ANYONE ELSE WHO HAS BEEN INVOLVED IN THE CREATION, PRODUCTION, OR DELIVERY OF THIS SOFTWARE SHALL BE LIABLE FOR ANY INDIRECT, CONSEQUENTIAL, OR INCIDENTAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE SUCH SOFTWARE EVEN IF MARSHALLSOFT COMPUTING, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES OR CLAIMS. IN NO EVENT SHALL MARSHALLSOFT COMPUTING, INC.'S LIABILITY FOR ANY SUCH DAMAGES EVER EXCEED THE PRICE PAID FOR THE LICENSE TO USE THE SOFTWARE, REGARDLESS OF THE FORM OF THE CLAIM. THE PERSON USING THE SOFTWARE BEARS ALL RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE.

Some states do not allow the exclusion of the limit of liability for consequential or incidental damages, so the above limitation may not apply to you.

This agreement shall be governed by the laws of the State of Alabama and shall inure to the benefit of MarshallSoft Computing, Inc. and any successors, administrators, heirs and assigns. Any action or proceeding brought by either party against the other arising out of or related to this agreement shall be brought only in a STATE or FEDERAL COURT of competent jurisdiction located in Madison County, Alabama. The parties hereby consent to in personam jurisdiction of said courts.

## 9 FCE Function Summary

Refer to the FCE Reference Manual (FCE\_REF) for detailed information on the FCE functions. A one-line summary of each function follows.

There are 34 functions in the FTP Client Component (FCE) library.

<code>fceAbort</code>	Abort FTP session.
<code>fceAttach</code>	Attach (initialize) FCE for use.
<code>fceByteToShort</code>	Converts from 8-bit ASCII characters to 16-bit.
<code>fceClose</code>	Close connection opened with <code>fceConnect</code> .
<code>fceCommand</code>	Sends arbitrary command to server [Rename,get status, etc]
<code>fceConnect</code>	Connect to FTP server.
<code>fceDelFile</code>	Delete file on server.
<code>fceDelServerDir</code>	Delete server directory.
<code>fceDriver</code>	FCE direct mode driver.
<code>fceErrorText</code>	Get text associated with error code.
<code>fceExtract</code>	Extract line from LIST buffer.
<code>fceFileLength</code>	Gets file length for file on server.
<code>fceGetFile</code>	Download (receive) file.
<code>fceGetInteger</code>	Get FCE integer parameter for FTP processing. [GET RESPONSE, CONNECT STATUS, BYTES SENT/RECEIVED, DATA PORT, etc.]
<code>fceGetList</code>	Get list of files on server.
<code>fceGetLocalDir</code>	Get local directory.
<code>fceGetLocalFList</code>	Gets list of files in up/download directory.
<code>fceGetLocalFSize</code>	Gets size of file in up/download directory.
<code>fceGetServerDir</code>	Get server directory.
<code>fceGetString</code>	Get FCE string parameter. [GET LINE COUNT, SERVER IP, LOCAL IP, FULL RESPONSE etc.]
<code>fceGetTicks</code>	Get system ticks.
<code>fceHello</code>	Send "NOP" command.
<code>fceIsConnected</code>	Get current connection status.
<code>fceMakeServerDir</code>	Create directory on server.
<code>fceMatchFile</code>	Match next filename in list.
<code>fcePutFile</code>	Upload (transmit) file to server.
<code>fceRelease</code>	Release FCE.
<code>fceSetInteger</code>	Set FCE integer parameter for FTP processing. [SET DATA PORT.FTP PORT, WRITE BUFSIZE, APPEND MODE, SLEEP TIME, HIDE PASSWORD, PASSIVE, MASTER INDEX, RESPONSE WAIT, etc.]
<code>fceSetLocalDir</code>	Set local directory.
<code>fceSetMode</code>	Set XFER mode (ASCII or Binary).
<code>fceSetServerDir</code>	Set server directory.
<code>fceSetString</code>	Set FCE parameter string. [SET LOG FILE, BIND TO LOCAL IP, WRITE TO LOG]
<code>fceShortToByte</code>	Converts from 16-bit ASCII characters to 8-bit.
<code>fceToInteger</code>	Converts ASCII text to integer.
<code>fceGetFileSize</code>	Get size of file on server.
<code>fceGetFileTime</code>	Get date & time stamp of file on server.

## 10 FCE Error Return Code List

The complete list of FTP Client Engine Component (FCE) error codes follows.

FCE_ABORTED	Internal checksum fails!
FCE_ACCEPT_SILENT	Timed out waiting for accept.
FCE_ALREADY_ATTACHED	Already attached.
FCE_BAD_KEY_CODE	Bad key code passed to fceAttach.
FCE_BAD_STATUS_FLAG	Bad status flag passed to fceStatus.
FCE_BUFFER_OVERFLOW	List buffer overflow.
FCE_BUFFER_TOO_SMALL	Buffer too small.
FCE_CANNOT_ALLOC	Cannot allocate memory.
FCE_CANNOT_COMPLY	Cannot comply.
FCE_CANNOT_CREATE_SOCKET	Cannot create socket.
FCE_CANNOT_OPEN	Cannot open file.
FCE_CHAN_OUT_OF_RANGE	Channel out of range.
FCE_CONNECT_ERROR	Error attempting to connect.
FCE_EOF	Socket has been closed.
FCE_EXPIRED	Evaluation version has expired.
FCE_FILE_IO_ERROR	File I/O error.
FCE_INVALID_SOCKET	Invalid socket.
FCE_IS_BLOCKING	WINSOCK is currently blocking.
FCE_LISTEN_ERROR	Listen error.
FCE_LISTENER_SILENT	No response on listener socket.
FCE_MODE_NOT_AB	Must specify 'A' or 'B' for mode.
FCE_NO_GREETING	Missing server greeting message.
FCE_NO_HOST	No host name.
FCE_NO_SERVER	Cannot find FTP server.
FCE_NO_SOCKET_ADDR	No available sockaddr structures.
FCE_NOT_ATTACHED	Must call fceAttach first.
FCE_NOT_COMPLETED	LIST/GET/PUT not completed.
FCE_NOT_SERVER	Illegal chars in server name.
FCE_PASS_NULL_ARG	PASSWORD not specified.
FCE_PASV_ERROR	Cannot find PASV port.
FCE_PORT_RANGE	Port number out of range.
FCE_SERVER_ERROR	FTP server returned error.
FCE_SERVER_NULL_ARG	SERVER not specified.
FCE_SOCKET_READ_ERROR	Socket read error.
FCE_SOCKET_WRITE_ERROR	Socket write error.
FCE_TIMED_OUT	Socket timed out.
FCE_USER_NULL_ARG	USER name not specified.

The numerical value for each error codes is listed in the file **fceErrors.txt**.

[ END ]