FTP Client Engine

Library for C/C++

Programmer's Manual

(FCE4C)

Version 4.0

October 10, 2023

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

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

Voice: 1.256.881.4630
Web: http://www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction Page 3
  1.1 Features Page 4
  1.2 Documentation Set Page 5
  1.3 Example Program Page 6
  1.4 Installation Page 7
  1.5 Uninstalling Page 7
  1.6 Pricing Page 7
  1.7 Updates Page 7
  1.8 Keycode (License Key) Page 7
2 Library Overview Page 8
  2.1 Dynamic Link Libraries Page 8
  2.2 Console Mode Page 8
  2.3 Using the Library Page 8
  2.4 Static Linking Page 9
  2.5 Win32 STDCALL and DECLSPEC Page 9
  2.6 Calling FCE from C++ Page 9
  2.7 Adding FCE To Your Existing Program Page 9
  2.8 Error Display Page 9
  2.9 Targeting a 64-Bit CPU Page 10
  2.10 64-bit FCE Page 10
3 Compiler Issues Page 11
  3.1 Compiling Using an IDE Page 11
  3.2 Command Line Tool Setup Page 12
  3.3 Command Line Batch Files Page 13
  3.4 Command Line Makefiles Page 14
4 Supported Compilers Page 15
  4.1 Microsoft Visual C/C++ Page 15
  4.2 Microsoft C++ .NET Page 18
  4.3 Microsoft C# Page 19
  4.4 Borland C/C++ Page 20
  4.5 Turbo C/C++ for Windows Page 20
  4.6 Borland C++ Builder Page 21
  4.7 Watcom C/C++ Page 21
  4.8 Lcc-Win32 C Page 22
  4.9 MinGW C Page 22
  4.10 Digital Mars C/C++ Page 22
5 Compiling Programs Page 23
  5.1 Compiling FCE Source Code Page 23
  5.2 Compiling Example Programs Page 24
6 Example Programs Page 26
  6.1 FCEVER Page 26
  6.2 GET Page 26
  6.3 WINFTP Page 26
  6.4 CONFTP Page 27
  6.5 BCB_PRJ Page 27
  6.6 MFC_PGM Page 27
  6.7 MULTI Page 27
  6.8 CONN Page 27
  6.9 MGET Page 28
  6.10 MPUT Page 28
  6.11 PROXY Page 28
  6.12 SPEED Page 28
  6.13 HELLO Page 28
  6.14 VC_VER Page 28
  6.15 CS_VER Page 28
  6.16 VC_GET Page 29
  6.17 CS_GET Page 29
  6.18 LoadLib Page 29
7 Revision History Page 30

1 Introduction

The **FTP Client Engine for C/C++ (FCE4C)** library is a toolkit that allows software developers to quickly develop FTP client applications in C/C++ and .Net.

The **FTP Client Engine (FCE)** is a component 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 quickly develop FTP software applications in C/C++ 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.

This **FTP Client Engine Programmers Manual** provides information needed to compile and run programs in a C/C++ programming environment.

FCE4C includes 17 C/C++ example programs, including Visual C++ and Visual Studio .NET, demonstrating FTP protocol processing. Microsoft Foundation Class (MFC) and Borland C++ Builder (BCB) and Visual C# examples are also provided.

The **FTP Client Engine for C/C++** component library supports and has been tested with C/C++, Microsoft Visual C++, Visual Studio .NET Framework (Visual C++ .NET, C# .NET), Visual C++ Express, Borland C/C++, Borland Turbo C++ for Windows, Borland C++ Builder, Watcom C/C++, LCC-Win32, MinGW and Digital Mars C compilers. **FCE4C** can also be used with most other C/C++ Windows compilers.

FCE4C runs under all versions of Windows (Windows 95, 98, ME, 2000, 2003, 2012, NT, XP, Vista, Windows 7 and 8). The **FTP Client Engine SDK** DLLs (FCE32.DLL and FCE64.DLL) can also be used from any language (Visual Basic, VB.NET, ACCESS, EXCEL, VBA, Borland Delphi, Visual FoxPro, COBOL, Xbase++, Visual dBase, Microsoft Office, etc.) capable of calling the Windows API.

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

1. FCE4C is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. Will run on machines without having .NET installed.
3. FCE4C does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
4. FCE is fully threadable.
5. The FCE functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the FTP Client Engine Library for Visual Basic (FCE4VB), Delphi (FCE4D), PowerBASIC (FCE4PB), Visual FoxPro (FCE4FP), dBASE (FCE4DB), and Xbase++ (FCE4XB). All versions of FCE use the same DLLs (FCE32.DLL and FCE64.DLL). However, the examples provided for each version are written for the specified computer programming language.

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:

- Supports 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.
- 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 also be called from managed code.
- Will run on machines with or without .NET installed.
- Works with all versions of Microsoft Visual C++ (V4.0 through Visual Studio 2022)
- Supports most Windows C/C++ compilers (Borland, Watcom, LCC-WIN32, Digital Mars, etc.).
- Can be used with Microsoft Visual Studio .NET and Visual C#
- Can be used with Microsoft Foundation Class (MFC), and Borland C++ Builder programs.
- 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 Basic, VB.NET, Visual FoxPro, Delphi, Xbase++, dBASE, COBOL, Access and Excel.
- Supports Windows XP through Windows 11..
- License covers all programming languages.
- Royalty free distribution with your compiled application.
- Can be purchased with or without source code
- Updates are free for one year (Updates to source code are separate).
- Evaluation version is fully functional.
- Unlimited one-year email and phone support.

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the first manual (FCE_4C) in the set.

- FCE4C Programmer's Manual (FCE_4C.PDF)
- FCE User's Manual (FCE_USR.PDF)
- FCE Reference Manual (FCE_REF.PDF)

The FCE_4C Programmer's Manual (FCE_4C.PDF) is the language specific (C/C++) 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 FTP in general as well as language independent programming issues such as technical support, 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 also be viewed online at

http://www.marshallsoft.com/fce4c.htm

1.3 Example Program

The following example demonstrates the use of some of the **FTP Client Engine** library functions:

```
#include <windows.h>
#include <stdio.h>
#include "fce.h"

void main(int argc, char *argv)
{int Code;
 // attach FCE
 Code = fceAttach(1, 0); // KEY_CODE = 0
 if(Code<0) ErrorExit(Code);
 // connect to server
 Code = fceConnect(0,(char *)"ftp.drivehq.com",
              (char *)"anonymous", (char *)"mike@gmail");
 if(Code<0) ErrorExit(Code);
 // change to proper directory
 Code = fceSetServerDir(0, (char *)"MarshallSoft/PublicFolder");
 if(Code<0) ErrorExit(Code);
 // set to ASCII xfer mode
 fceSetMode(0,'A');
 // download the file
 Code = fceGetFile(0,(char *)"fce-new.txt");
 if(Code<0) ErrorExit(Code);
 // QUIT */
 fceClose(0);
 fceRelease();
}
```

In the example program above, **fceConnect** is called to connect to the FTP server as user "anonymous" and password "msc@traveller.com".

The server directory is changed to "pub/oem", the transfer mode is set to ASCII, and the file "fce-new.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 FCE4C, a Windows C/C++ compiler should already be installed and tested. In particular, include command line tools when installing a compiler to be to use command line makefiles. Refer to the file, MAKEFILE.TXT, for help with makefiles.

(2) Unzip FCE4C40.ZIP (evaluation version) or FCExxxxx.ZIP (registered version; xxxxx is the Customer ID).

(3) Run the installation program SETUP.EXE which will install all FCE4C files, including copying FCE32.DLL (and FCE64.DLL) to the Windows directory.

All recent Win32 C/C++ compilers support the "declspec" keyword. Microsoft VC (version 4.0 and up), Digital Mars, Watcom (version 11.0 and up), MinGW GCC, and LCC-Win32 compilers support the "declspec" keyword.

1.5 Uninstalling

Uninstalling FCE4C is very easy. First, delete the FCE project directory created when installing FCE4C. Second, delete FCE32.DLL and FCE64.DLL from the Windows directory, typically C:\

1.6 Pricing

A developer license for FCE4C can be registered for $119 ($299 with ANSI C source code to the DLL). Purchasing details can be found in Section 1.3, "How to Purchase", of the FCE User's Manual (fce_usr.pdf).

1.7 Updates

When a developer license is purchased for FCE, the developer will be sent a registered DLL plus a license file (FCEExxxx.LIC). 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/update.htm

After one year, your license must be updated if you want to be able to download updates. Your license can be updated for $33 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 $77.

Note that the registered DLLs, (FCE32. DLL and FCE64.DLL) never expire. If source code was previously purchased, updates to the source code can be purchased for $100 along with the license update.

1.8 Keycode (License Key)

The FCE DLLs have a keycode encoded within them. The keycode is a 9 or 10-digit decimal number (unless it is 0), and will be found in the file KEYCODE.H. The keycode for the evaluation version is 0. The developer will receive a new key code after registering. The KEYCODE is passed to **fceAttach**.

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 registering, it is best to remove the evaluation version of the FCE DLL from the Windows search.

2 Library Overview

2.1 Dynamic Link Libraries

The **FTP Client Engine Library SDK** is implemented as a dynamic link library (DLL). Both a Win32 DLL and Win64 DLL are included. 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 DLL's have over other "popular" library formats such as VBX or OCX is that DLL's are callable by all Windows applications. Since DLL's are the building blocks of the Windows Operating System, they will not be replaced by a "newer technology".

2.2 Console Mode

FCE library functions can be called from Win32 console mode programs. A "console mode" program is a Windows 95/98/NT/2000/2003/Me/XP/Vista/7&8 Win32/Win64 command line program running in a command window. Although console mode programs look like DOS programs, they are Win32 (or Win64) programs that have access to the Win32 (or Win64) API and the entire Windows address space. Programming using console mode programs reduces the complexity of using GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code.

It is recommended that console mode programs be run in window mode rather than full screen mode. To change Console Mode windows to "windows mode" from "full screen" mode:

Start a command window: Click Windows "Start Button", then "run", then enter

```
\windows\system32\cmd.exe
```

(followed by a carriage return) into the Run window. When the command window comes up, right click on the blue menu bar at the top of the menu, then click "Defaults", then choose the "Options" tab. Click the "Window" option under "Display Options".

2.3 Using the FCE Library

The **FTP Client Engine** component library has been tested on multiple computers running Windows 95/98/Me/NT/2000/2003/2012/XP/Vista/7/8 and Windows NT/2000.

The FCE4C library has been tested with several C/C++ compilers, including Microsoft Visual C++ (all versions including Visual Studio .NET and Visual Studio C#), Borland C/C++, Borland C++ Builder, Turbo C/C++ for Windows, Watcom C/C++, MinGW and Digital Mars C/C++.

The SETUP installation program will copy the Lib's and DLL to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the FCE4C files are copied to the directory specified (default \FCE4C). Three sub-directories are created, as follows:

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

2.4 Static Linking

Static libraries require the object file version (fce*.obj) of the DLL or source code. To create an application that links FCE32.OBJ (or FCE64.OBJ) statically:

(1) All application code that includes FCE.H must define STATIC_LIBRARY before including FCE.H
(2) Your application must link with WSOCK32 (for WIN32).

If using Microsoft Developer Studio, make these changes:

(1) To your project file: Do NOT add FCE 32.LIB to your project file.
(2) To the settings: (See "Build Settings" or "Project/Settings")
(2a) C/C++ Tab: Add STATIC_LIBRARY to "preprocessor definitions:"
(2b) Link Tab: Add WSOCK32.LIB and FCE32.OBJ to "object/library modules:"
(3) Add #include "FCE.H" to all source files that make calls to FCE functions.

2.5 Win32 STDCALL and DECLSPEC

Our 32-bit libraries are compiled using the _stdcall and _declspec keywords. This means that they use the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are no leading underscores or trailing "@size" strings added to function names.

Microsoft Visual C/C++ users can look at the DLL function names using the dumpbin.exe executable:

dumpbin /exports fce32.dll

2.6 Calling DLL Functions from C++

Like Windows itself, FCE functions are coded in ANSI C, but they can be called directly from both ANSI C programs and from C++ programs.

FCE functions can also be called using the C++ class wrapper **ffce**. Refer to **ffce.cpp** and **ffce.h** as well as the HELLO.CPP program for an example.

2.7 Adding FCE DLL Functions To An Existing Program

In order to call FCE functions from an existing program,

(1) Add

```
#include "fce.h"
```

to the application source code (use fce.h1 instead of fce.h for Borland programs),

(2) Link with FCE32.LIB (for MSVC), FCE32BCB.LIB (Borland C/C++ and C++ Builder), FCE32.LIB (Watcom), or FCE32LCC (Win32/LCC)

and recompile from source.

Link Win64 with FCE64.LIB rather than FCE32.LIB. Also refer to Section 4.0 below.

2.8 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.

2.9 Targeting a 64-Bit CPU

If a compiler generates 32-bit application code and is running on a 64-bit version of Windows, then compiling and linking is the same as if it were on a 32-bit Windows system. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

If a compiler generates 64-bit application code and is running on a 64-bit version of Windows, then the compiler must be reconfigured to generate 32-bit application code if the application will call 32-bit DLL's such as FCE32.DLL. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

2.9.1 Visual Studio C/C++: Versions 2005 through 2022

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click the "Configuration Manager" button in upper right corner.
Click the drop-down button below "Platform".
Click <New...>, then choose "x86" (Win32).

2.9.2 Visual Studio Visual Basic: Versions 2005 through 2022

With a project selected in Solution Explorer, on the Project menu, click Properties.
Click "Build", then "Configuration Manager".
Click the drop-down button below "Active Solution Platform".
Click <New...>, then change "Any CPU" to "x86".

2.10 64-bit FCE

64-bit DLL's may only be used by 64-bit application programs running on 64-bit Windows computers. This means that 64-bit application programs must be linked with FCE64.LIB instead of FCE32.LIB.

However, if a compiler generates 32-bit code, the application must be linked with FCE32.LIB even though it may be running on a 64-bit machine.

3.0 Compiler Issues

Application programs can be compiled using an IDE or command line compiler tools. The following sections provide general compiler information.

3.1 Compiling Using an IDE

All current windows compilers have an "Integrated Development Environment" (IDE) for building application programs in the Windows environment.

Note that not only do IDE's vary between the different compiler manufacturers, but they also vary from version to version for the same compiler.

3.1.1 Compiling Example Programs with an IDE

Most of the example programs can be compiled from your compiler's IDE. For Visual C++, "project makefiles" are used since they can be used by all versions of Visual C++. When opening the workspace, select "makefiles(.mak)" for the file type.

Alternatively, for VC++ v6.0, select "projects (.dsp)" for the file type.

3.1.2 Compiling New Projects with an IDE

All of the IDE's use the concept of a file hierarchy. For example, the FCEVER example program file hierarchy in the IDE (for 32-bit) should look like:

```
FCEVER.EXE
   +++ FCEVER.C
   +++ FCE32.LIB
```

Replace FCE32.LIB with FCE32BCB.LIB if using Borland C++ Builder, and with FCE32LCC.LIB if using LCC-Win32.

The order of the files is not significant. Refer to Section 4 below for a particular IDE.

3.2 Command Line Tool Setup

Many software developers overlook the power of using command line compilers. There are a number of very significant advantages to using the command line version of your C/C++ compiler. Among these are:

- **Easy of Use**: Once set up, typing a single key can compile one or a thousand programs.
- **Power**: Using the command line allows the use of batch files, automating complicated builds.
- **Compatibility**: Command line makefiles (unlike IDE project files) are normally compatible from one version of your compiler to the next.

If you want to compile from the command line, your command line compiler tools must be set up properly. Note that you have an option of installing the command line tools (or not) when your compiler is first installed. Refer to your compiler manufacturer's manual for details.

If necessary, you can increase the size of your environment table space by adding

```
SHELL=C:\COMMAND.COM /e:1024 /p
```

to CONFIG.SYS in C:\ and then rebooting. Yes, this works for all versions of Windows, including Windows NT, 2000, XP, 7, 8 and Vista.

For all compilers, your path should point to the compiler's BIN directory. For example, to add "C:\BC50\BIN" to your existing path, use

```
PATH C:\BC50\BIN;%PATH%
```

3.2.1 Microsoft Visual C++

Set LIB and INCLUDE environment variables. For example,

```
SET INCLUDE=C:\MSVC\INCLUDE
SET LIB=C:\MSVC\LIB
```

3.2.2 Borland

Check that TURBOC.CFG, BCC32.CFG, TLINK.CFG, and TLINK32.CFG all have the correct information in them, as they should have when your compiler was installed. For example,

```
-IC:\BC5\INCLUDE
-LC:\BC5\LIB
```

BRCC (the Borland Resource Compiler) doesn't use the *.CFG files. Set the INCLUDE environment variable or BRCC will not be able to find the INCLUDE files (such as WINDOWS.H). For example,

```
SET INCLUDE=C:\BC5\INCLUDE
```

Clear the LIB environment variable (so it is not present when SET is typed at the command line) with

```
SET LIB=
```

Note that programs compiled with Borland C/C++ include the file fce.h1 rather than fce.h.

3.2.3 Watcom

Set the WATCOM environment variables to point to your compilers include (H) and BIN directories. For example,

```
SET INCLUDE=C:\WC11\H;C:\WC11\H\NT
SET WATCOM=C:\WC11
SET EDPATH=C:\WC11\EDDAT
SET WWINHELP=E:\BINW
```

3.2.4 LCC-Win32

The LCC environment variables are set like the others. For example,

```
SET INCLUDE=C:\LCC\INCLUDE
SET LIB=C:\LCC\LIB
```

After making the above changes for your compiler, type PATH at the command line prompt to verify the search path, and type SET at the command line prompt to verify the INCLUDE and LIB environment variables.

3.2.5 MinGW GCC

The MinGW environment variables are set like the others. For example,

```
SET INCLUDE=C:\MinGW\INCLUDE
SET LIB=C:\MinGW\LIB
```

After making the above changes for your compiler, type PATH at the command line prompt to verify the search path, and type SET at the command line prompt to verify the INCLUDE and LIB environment variables.

3.2.6 Digital Mars

The Digital Mars environment variables are set like the others. For example,

```
SET INCLUDE=C:\DM857\INCLUDE
SET LIB=C:\DM857\INCLUDE
```

After making the above changes for your compiler, type PATH at the command line prompt to verify the search path, and type SET at the command line prompt to verify the INCLUDE and LIB environment variables.

3.3 Command Line Batch Files

If your compiler installation includes command line tools, then all of the example programs can be compiled directly from the command line. These same compiler commands can also be placed in a batch file.

3.4 Command Line Makefiles

Command line makefiles originated on UNIX systems. They are the standard way that C/C++ programs are constructed in command line environments. The advantage of makefiles (as compared to an integrated development environment) is that all compiler switches are always coded within the makefile and the makefile can be run with a single keystroke.

Command line makefiles are provided for Microsoft, Borland, Watcom, and LCC-Win32 command line compilers. They can be found in the APPS sub-directory:

```
borland50.zip     Borland C/C++ 5.0 makefiles.
borland55.zip     Borland C/C++ 5.5 makefiles.
lcc.zip           LCC-WIN32 project command files.
microsoft60.zip   Microsoft C/C++ (v1.52, v4.0, v5.0, v6.0) makefiles.
watcom11.zip      Watcom C/C++ 11 makefiles.
gcc.zip           MinGW GCC project command files.
dm.zip            Digital Mars project make files.
```

4.0 Supported Compilers

Our libraries have been tested with Microsoft Visual C++ (all versions including Visual Studio C++ and Visual Studio C# through Visual Studio 2013), Borland C/C++, Borland/Embarcadero C++ Builder, Borland Turbo C/C++, Watcom C/C++, Lcc-Win32, MinGW GCC and Digital Mars. Other Windows C/C++ compilers may work as well. Other Windows C/C++ compilers may work as well. Refer also to Section 5, "Compiling Example Programs".

4.1 Microsoft Visual C++ (all versions)

Microsoft Visual C/C++ programs can be compiled from either the command line or from within the Microsoft Visual Studio development environment.

The last Win16 Microsoft compiler is version 1.52. All Microsoft Visual C++ compilers (beginning with version 4.0) compile Win32 programs ONLY. FCE does not support Win16.

4.1.1 Microsoft Command Line Makefiles

Programs can be compiled using command line makefiles. All Microsoft Win32 command line makefiles end with "'_m_". To compile using a makefile, use the Microsoft NMAKE utility. For example,

```
NMAKE -f FCEVER._M_
```

FCE can be used with Microsoft Foundation Class (MFC) programs.

The file MICROSOFT60.ZIP contains the Microsoft C/C++ (v4.0, v5.0, v6.0) command line makefiles.

4.1.2 Microsoft Developer Studio (VC v4.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK" , such as FCEVER.MAK

To create a new project in MSVC v4.0, choose "File", then "New", then "Project Workspace". Select "Application" or "Console Application" for "Type:" and your project name for "Name:". Choose Win32 for platform. Then select "Create". Select "Insert", then "Files into Project". Add all filenames including any resource file (.RC) and FCE32.LIB. Lastly, select "Build", then "Rebuild All". Be sure to specify /YX rather than /Yu in your project settings [Build, Settings…, C/C++].

4.1.3 Microsoft Visual Studio (VC v5.0)

To open an existing project, choose "File", then "Open Workspace", and then select "makefiles" from the list of file types. Most of the example programs have Microsoft Developer C/C++ project makefiles, ending with ".MAK", such as FCEVER.MAK.

To create a new project in MSVC v5.0, choose "File", then "New", then "Win32 Application" or "Win32 Console Application" and your project name. Check "Create new workspace". Select "Project", then "Add to Project". Add all filenames including any resource file (.RC) and FCE32.LIB. Lastly, select "Rebuild All".

If the compiler complains that it cannot find "_main", "Console Application" was chosen but the program being compiled is a GUI application. If the compiler complains that it cannot find "WinMain", "Application" was chosen but the program being compiled is a Console Mode application. Be sure to specify /YX rather than /Yu in your project settings [Build, Settings…, C/C++].

4.1.4 Microsoft Visual Studio (VC v6.0)

To open an existing project, follow the same directions as for MSVC v5.0, except that a DSP project file may be used instead of the MAK project makefile.

To create a new project in MSVC v6.0, follow the same directions as for MSVC v5.0 above.

4.1.5 Microsoft Visual Studio 2003 through 2008 (VC++ 7.0, VC++ 8.0 and VC++ 9.0)

Open the VC project file (files ending in "`.vcproj`" or "`.dsp`"), build and run.

The 32-bit Visual Studio project files are contained in VS2008(32bit)vcproj.zip and the 64-bit Visual Studio project files are contained in VS2008(64bit)vcproj.zip

4.1.6 Microsoft Visual Studio 2010

Visual Studio project file names changed from "`.vcproj`" to "`.vcxproj`" beginning with Visual Studio 2010.

The 32-bit Visual Studio project files are contained in VS2010(32bit)vcxproj.zip and the 64-bit Visual Studio project files are contained in VS2010(64bit)vcxproj.zip

Open the VC project file (files ending in ".vcxproj"), build and run, as in previous version of Visual Studio.

4.1.7 Microsoft Visual Studio 2012

The 32-bit Visual Studio project files are contained in VS2012(32bit)vcxproj.zip and the 64-bit Visual Studio project files are contained in VS2012 (64bit)vcxproj.zip

Open the VC project file (files ending in ".vcxproj"), build and run, as in previous version of Visual Studio.

In order to convert a VS 2010 project file to VS 2012:

1. Open the VS 2010 project file with VS 2012.
2. Let VS 2012 update to 2012 format when prompted.
3. Select "Project", "Properties", "Linker", then "Advanced".
4. Change the "Image Has Safe Exception Handlers" to NO (/SAFESEH:NO)
5. Save project.

The above will insert the line

```
<ImageHasSafeExceptionHandlers>false</ImageHasSafeExceptionHandlers>
```

into your project file as the last line before `</Link>`

Unless you require a 16-byte character set (unicode), change the project file from

```
<CharacterSet>MultiByte</CharacterSet>
```

to

```
<CharacterSet>NotSet</CharacterSet>
```

4.1.8 Microsoft Visual Studio 2013 Thru 2022

The 32-bit Visual Studio project files are contained in

vs2013(32bit)vcxproj.zip and vs2013(64bit)vcxproj.zip.
vs2015(32bit)vcxproj.zip and vs2015(64bit)vcxproj.zip.
vs2017(32bit)vcxproj.zip and vs2017(64bit)vcxproj.zip.
vs2019(32bit)vcxproj.zip and vs2019(64bit)vcxproj.zip.
vs2022(32bit)vcxproj.zip and vs2022(64bit)vcxproj.zip.

Open the VC project file (files ending in ".vcxproj"), build and run, as in previous version of Visual Studio.

Also see the section of Visual Studio 2012.

4.1.9 Microsoft C++ Express Edition (VC++ 9.0)

The "Express Edition" of Microsoft C++ 9.0 is available as a free download at
http://www.microsoft.com/express/download/

Open the VC project file (files ending in ".vcproj" or ".dsp"), build and run, as in previous versions of Visual Studio. Also see Section 4.2 below.

4.2 Microsoft Visual C++ .NET

All Visual Studio C++ .Net projects end with extension ".vcproj". For example,

```
vc_vers.vcproj
```

In order to open an existing Visual C++ .Net project, choose "File", "Open", and then "Project" from the Visual Studio menu. Specify the directory containing the Visual C++ .Net project files (for example, C:\FCE4C\APPS).

In order to call FCE functions from Visual C++ .Net programs, do the following to your existing Visual C++ .Net project:

(1) Add

```
#include "fce.h"
#include "keycode.h"
```

after your existing #include statements.

(2) Open the project properties window under "Project" on the main menu. If your project is named "MyProject", then select "MyProject properties".

(3) Select "Configuration Properties", "Linker", "Input", "Additional Dependencies", and then type in "FCE32.LIB" into the edit box. Note that FCE32.LIB must be in your project directory along with all of your source files.

(4) Rebuild.

NOTE: If using pre-compiled headers, the include statement

```
#include "stdafx.h"
```

must be the first include statement in your program.

4.3 Microsoft Visual C#

FCE functions can be called from C# (C-sharp) in the same manner as Win32 API functions.

All C# projects end with extension ".csproj". For example,

    cs_vers.csproj

In order to open an existing C# project, choose "File", "Open", and then "Project" from the Microsoft Development Environment. Specify the directory containing the C# project files (for example, C:\FCE4C\APPS).

In order to call FCE functions from your C# programs, do the following to your existing C# source code:

Add the contents of file fce_funs.cs to source code after

    public class fce : System.Windows.Forms.Form

Add the constants from fce_sons.cs to your program as they are needed.

Look at the cs_vers program in the APPS directory for an example.

4.4 Borland C/C++

Borland C/C++ version 5.0 programs can be compiled from either the command line (using makefiles ending with "._b_") or from within the Borland development environment using Borland v5.0 or above.

Borland C/C++ Version 5.5 (which can be downloaded from www.borland.com) is a free Win32 console mode compiler (no IDE). Makefiles for BC v5.5 end with "._i_", and (like Borland C++ Builder) use ILINK32 rather than TLINK32. Be careful with linker response files (*.RSP) -- they must **NOT** end with a carriage return / line feed!

Borland programs always link with FCE32BCB.LIB.

4.4.1 Borland Command Line Makefiles

Programs can be compiled using command line makefiles. All Borland Win32 command line makefiles end with "._b_" (or "._i_" for Borland_5.5). To compile using a makefile, use the Borland MAKE utility. For example,

```
MAKE -f  FCEVER._B_
```

The file, BORLAND50.ZIP, contains the Borland C/C++ 5.0 command line makefiles, and the file, BORLAND55.ZIP, contains the Borland C/C++ 5.5 command line makefiles.

4.4.2 Borland IDE

To create a new project, first turn off LINKER case sensitivities: Choose "Options", "Projects", "Linker", "General". Turn off the "case sensitive link" and "case sensitive exports and imports" boxes.

Next, choose "Files", then "New Project". Use the INS (Insert) key to pop up a dialog box into which the project file names are entered. Lastly, add FCE32BCB.LIB to your project. FCE32BCB.LIB can also be created from FCE32.DLL using the Borland IMPLIB utility:

```
IMPLIB FCE32BCB.LIB FCE32.DLL
```

Select "GUI" or "Console" for the "Target Model:". Only "Static" or "Dynamic" should be checked for "Standard Libraries:"

NOTE1: If, after linking in the IDE, you get unresolved external references to the library functions in which each function name is all upper case, then you have NOT turned off case sensitivity as described above.

NOTE2: If you get errors compiling the windows header file "WINDOWS.H", turn on "Borland Extensions" in "Options", "Project", "Compiler", "Source".

4.5 Borland Turbo C/C++ for Windows

Borland Turbo C/C++ for Windows does not have command line tools, so all programs must be compiled from the Turbo C/C++ integrated environment.

Follow the same directions as above (Borland IDE), except that the "Target Model:" can be any listed.

4.6 Borland C++ Builder

Borland C++ Builder does not have command line tools, so all programs must be compiled from the Borland C++ Builder integrated environment. Compile the BCB example program BCB_PRJ with BCB_PRJ.MAK if running BCB version 1 through 3, and compile with BCB_PRJ.BPR if running BCB version 4 or above.

To load the BCB_PRJ example project, Choose "File" / "Open Project" on the menu bar. Load BCB_PRJ.MAK (or BCB_PRJ.BPR). Then, choose "Build All" from "Project" to create the executable.

Note that FCE32BCB.LIB is the LIB file used with Borland C++ Builder. FCE32BCB .LIB can be created from FCE32.DLL by using the Borland IMPLIB program:

```
IMPLIB FCE32BCB.LIB FCE32.DLL
```

The file C-BUILDER.ZIP contains the Borland C++ Builder project makefiles.

4.7 Watcom C/C++

FCE4C works with Watcom 11 and Open Watcom (http://www.openwatcom.org).

Watcom C/C++ programs can be compiled from either the command line or from within the Watcom development environment.

4.7.1 Watcom Command Line Makefiles

Win32 programs can be compiled using command line makefiles. All Watcom command line makefiles end with "._w_" for Win32 makefiles. To compile using a makefile, use the Watcom WMAKE utility. For example,

```
WMAKE -f FCEVER._W_
```

Win32 programs can also be compiled using command line batch files. See FCEVER$.BAT for an example of a console mode command line batch file and FCEVER$.BAT for an example of a GUI mode command line batch file. To run these command line batch files from the command line, type

```
FCEVER
```

The file, WATCOM11.ZIP, contains the Watcom C/C++ 11 command line makefiles.

4.7.2 Watcom IDE

To create a new project, choose "File", then "New Project". Enter the project name and then choose Win32 as the target. Use the INS (Insert) key to pop up a dialog box into which the project file names are entered.

Select "Options" from the main window, then "C Compiler Switches", then "10". Memory Models and Processor Switches". Check "80386 Stack based calling [-3s]", then check "32-bit Flat model [-mf]".

4.8 LCC-Win32 C

Lcc-Win32 C/C++ programs can be compiled from either the command line or from within the development environment.

Lcc-Win32 is a freeware C compiler developed and distributed by Jacob Navia at

http://www.cs.virginia.edu/~lcc-win32/

To use our DLL's with Lcc-Win32, you must link with FCE32LCC.LIB. This file can also be re-created using the Lcc-Win32 utility BUILDLIB.

```
buildlib fce32.lcc fce32lcc.lib
```

Then, compile and link as normal. For example, to compile the FCEVER example program,

```
lcc -DWIN32 fcever.c
lcclnk fcever.obj fce32.lib -subsystem:console
```

To compile the GUI mode example VERS,

```
lcc -DWIN32 vers.c
lcc -DWIN32 paint.c
lrc vers.rc
lcclnk vers.obj paint.obj fce32lcc.lib vers.res -subsystem:windows
```

The file, lcc-win32.zip, contains the LCC-WIN32 project command files.

4.9 MinGW C

MinGW (Minimalist GNU for Windows) is part of the GNU Compiler Collection (GCC), and GNU Binutils, for use in the development of native Microsoft Windows applications. See http://www.mingw.org

Console mode programs are compiled from the command line; for example

```
gcc -Wall fcever.c fce32.lib -o fcever.exe
```

The current version of MinGW does not come with a resource compiler (for RC files), so GUI Window applications that use resource files cannot be compiled with MinGW.

The file, gcc.zip, contains the MinGW (gcc) project command files.

4.10 Digital Mars

Digital Mars C/C++ (see http://www.digitalmars.com) programs can be compiled from the command line by using the Digital Mars "make" program. All Digital Mars command line makefiles end with "_d_". For example,

```
make -f fcever(s)._d_
```

The file dm.zip contains all Digital Mars makefiles.

5 Compiling Programs

Microsoft Visual Studio currently is the only C/C++ compiler that can generate 64-bit application code.

5.1 Compiling FCE Source Code

This section applies only to those who have purchased source code for the **FTP Client Engine Library.**

FCE32.DLL has been compiled using Microsoft Visual C++, and is callable from applications written using Microsoft, Borland, or Watcom compilers. If FCE32.C is recompiled using Borland or Watcom compilers, then the resulting FCE32.DLL can only be used by applications compiled with the same compiler, unless the "_stdcall" and "_declspec" keywords are specified.

Microsoft Visual C++ is used to create FCE32.DLL and FCE32.OBJ (for static linking).

In order to create FCE32.DLL, type:

```
NMAKE -f FCE32._M_
```

In order to create FCE32.OBJ (for static linking) type

```
NMAKE -f FCE32S._M_
```

Alternatively, FCE32.C can be included in a project (along with MSC-VS.C and MSC-STR.C) like any other C file. Before compiling, define the symbol STATIC_LIBRARY.

FCE64.DLL has been compiled using Microsoft Visual Studio 2008, and is callable from 64-bit applications programs.

5.2 Compiling Example Programs

The example programs can be compiled by using either the command line compiler or the compiler integrated development environment (IDE). Most compiler vendors provide both IDE and command line tools, although some compilers are command line only (Borland C/C++ 5.5 and LCC-Win32) or IDE only (Borland C/C++ Builder). Also see Section 4, "Supported Compilers".

5.2.1 Compiling Using Visual C++ (VC v4.0, v5.0, and v6.0)

Microsoft Visual C/C++ (v4.0, v5.0, v6.0) compiles only 32-bit programs.

Programs can be compiled with Microsoft Visual C/C++ using either Developer Studio / Visual Studio or the command line compiler. Project makefiles (files ending in ".MAK") are provided for Developer Studio / Visual Studio. Command line makefiles (files ending in "._M_") are provided for use with the command line compiler (e.g.: NMAKE -f FCEVER._M_).

MSVC v6.0 project files (files ending in ".DSP") are also provided.

5.2.2 Compiling Using Visual C++ .Net

Microsoft Visual C/C++ .NET compiles only 32-bit programs.

Programs can be compiled with Microsoft Visual C/C++ .NET using either Visual Studio .NET or the command line compiler. Project files (files ending in ".VCPROJ") are provided for Visual Studio (e.g.: VC_VERS.VCPROJ).

Command line makefiles (files ending in "._M_") are provided for use with the Visual C++ .NET command line compiler (e.g.: NMAKE -f FCEVER._M_).

5.2.3 Compiling Using Visual C#

FCE functions can also be called from C# programs in the same manner in which Win32 API DLL functions are called.

5.2.4 Compiling Using Borland C/C++ 5.0

Borland C/C++ 5.0 can compile both 32-bit and 16-bit programs.

Programs can be compiled with Borland C/C++ using either the Borland IDE or the command line compiler. Several Project files (files ending in ".IDE") are provided for the Borland IDE (unzip BC50-IDE.ZIP) and command line makefiles (files ending in "._B_") for the Borland command line compiler. For example (MAKE -f FCEVER._B_)

5.2.5 Compiling Using Borland C/C++ 5.5

Borland C/C++ 5.5 is a command line compiler that can compile both 32-bit programs only. Command line makefiles (ending in "._I_") are provided, e.g., MAKE -f FCEVER._I_.

5.2.6 Compiling Using Borland C++ Builder

Borland C++ Builder (BCB) is an IDE that features "drag and drop" forms building (like Delphi and Visual Basic).

BCB compiles 32-bit programs only. Compile the BCB example program BCB_PRJ with BCB_PRJ.MAK if running BCB version 1 through 3, and compile with BCB_PRJ.BPR if running BCB version 4 or above.

The file C-BUILDER.ZIP contains the Borland C++ Builder project makefiles.

5.2.7 Compiling Using Watcom 11 / Open Watcom

Programs can be compiled with Watcom 11 using either the Watcom IDE or the command line compiler. Several command line makefiles (files ending in "._W_") for the Watcom command line compiler. For example (e.g.: WMAKE -f FCEVER._W_)

5.2.8 Compiling Using LCC-WIN32

LCC-Win32 compiles only 32-bit C (not C++) programs.

Command line batch files (ending with "$.bat) are provided for several of the projects. For example, to compile FCEVER using LCC-Win32, type FCEVER$.BAT on the command line.

5.2.9 Compiling Using MinGW

Command line batch files (ending with "$GCC.bat) are provided for several of the projects. For example, to compile FCEVER using MinGW (GCC), type FCEVER$GCC.BAT on the command line.

5.2.10 Compiling Using Digital Mars

Programs can be compiled with Digital Mars. Command line makefiles (files ending in "._d_") are provided. For example,

```
make -f fcever(s)._d_
```

6 Example Programs

Some of the example programs are written in GUI mode (WINFTP, VC_GET, BCB_PRJ), although most are written in Win32 console mode. Note that console mode programs must be run from the Windows command prompt. Also note that console mode programs can be converted to GUI mode by adding the necessary Windows interface code.

Please **read the comments in the example program source code** before compiling!

Makefiles are classified as follows:

- *._m_ Microsoft C/C++ makefile (command line).
- *._b_ Borland C/C++ makefile (command line).
- *._w_ Watcom C/C++ makefile (command line).
- *._d_ Digital Mars C/C++ makefile (command line).

Files ending with .MAK (and .BPR) include:

- FCEVER32.MAK Microsoft Visual C/C++ Developer Studio makefile.
- WINFTP32.MAK Microsoft Visual C/C++ Developer Studio makefile.
- MFCPGM32.MAK Microsoft Foundations Class (MFC) makefile (Win32).
- BCB_PRJ.MAK Borland C++ Builder makefile (BCB version 1 through 3).
- BCB_PRJ.BPR Borland C++ Builder makefile (BCB version 4 and above).

6.1 FCEVER

The first example program is the console mode program FCEVER (FCE Version) that displays the FCE library version number and registration string.

There are command line makefiles for Microsoft (FCEVER32._M_), Borland (FCEVER32._B_), and Watcom (FCEVER32._W_), as well as a Microsoft Developer Studio makefile (FCEVER32.MAK).

After compiling, from the command line, type:

```
FCEVER
```

6.2 GET

GET is a console mode Win32 FTP client application that connects to our FTP server anonymously and downloads the file “fce-new.txt”. After compiling, from the command line, type:

```
GET
```

6.3 WINFTP

WINFTP is a Win32 GUI (Graphical User Interface) application that can be used to connect to a FTP server and upload, download, and delete files. See WINFTP.TXT in the DOCS directory for more details.

There are command line makefiles for Microsoft (WINFTP32._M_), Borland (WINFTP32._B_), Watcom (WINFTP32._W_), and Microsoft Developer Studio makefile (WINFTP32.MAK). There is also an Lcc-Win32 compiler batch file (WINFTP$.BAT).

6.4 CONFTP

CONFTP is a generic FTP client (console mode) application that exercises most of the FCE functions. After compiling, start CONFTP from the command line

```
CONFTP server user password
```

6.5 BCB_PRJ

BCB_PRJ is a Borland C++ Builder (BCB) example program similar to WINFTP.

For Borland C++ Builder version 1 through 3, use project file BCB_PRJ.MAK. For Borland C++ Builder version 4 and above, use project file BCB_PRJ.BPR.

BCB programs must always link with FCE32BCB.LIB rather than with FCE32.LIB.

6.6 MFC_PGM

MFC_PGM is a Microsoft Foundation Class equivalent of the GETPRO example program. Its purpose is to demonstrate calling FCE functions from MFC. Compile from the command line with MFCPGM32.MAK.

6.7 MULTI

MULTI is a Win32 console mode application that uses multiple threads in order to log on to several FTP servers concurrently.

Be sure to edit the FTP account information in MULTI.C before compiling.

6.8 CONN

CONN is a console mode program used to test connection timeouts. For example, to connect to server "ftp.hiwaay.com" as user "anonymous" with password "you@yourisp.net" waiting for a maximum of 5 seconds for the connection to complete, type (at the command line):

```
CONN ftp.hiwaay.net anonymous you@yourisp.net 5
```

6.9 MGET

MGET is a console mode program that downloads all files (from the server directory) that match a user specified file specification using ? and * wildcard characters.

Edit the lines following "*** PROGRAMMER" in MGET.C before compiling. See the COMPILE comment in MGET.C for information on compiling.

6.10 MPUT

MGET is a console mode program that uploads all files (to the server directory) that match a user specified file specification using ? and * wildcard characters.

Edit the lines following "*** PROGRAMMER" in MPUT.C before compiling. See the COMPILE comment in MPUT.C for information on compiling.

6.11 PROXY

PROXY is a console mode program that connects to a FTP server through a proxy server using the “PROXY USER” protocol. 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.pdf).

Be sure to edit the FTP account information in PROXY.C before compiling.

6.12 SPEED

SPEED is a console mode program designed to test download and upload speeds. Download a file greater than one megabyte for the most accurate results.

6.13 HELLO

HELLO is a console mode program similar to FCEVER (FCE Version) that demonstrates how to use the 'ffce' C++ class.

6.14 VC_VER

VC_VER is a Microsoft Visual Studio (C.NET) version of the FCEVER (FCE Version) example program.

6.15 CS_VER

CS_VER is a Microsoft Visual Studio C# version of the FCEVER (FCE Version) example program.

6.16 VC_GET

VC_GET is a Microsoft Visual Studio (C.NET) version of the GET example program.

6.17 CS_GET

CS_GET is a Microsoft Visual Studio C# version of the GET example program.

6.18 LoadLib

LoadLib is a console mode program that demonstrates how to dynamically load FCE32.DLL functions from a user specified directory.

7 Revision History

Version 1.0 "Beta": February 12, 1999.

- The Beta release.

Version 1.0: March 22, 1999

- The initial release.

Version 1.1: June 14, 1999

- Added fceMakeServerDir and fceDelServerDir
- Added PASSIVE mode to make FCE firewall safe.
- Allow filename or directory name in fceGetList.

Version 1.2: August 10, 1999

- fceHello function added.
- Improved example programs.
- Error message return on warning levels.
- Several (relatively obscure) bugs corrected.

Version 2.0: April 24, 2000.

- WriteBufferSize default reduced to 128.
- Added FCE_GET_LINE_COUNT to fceGetString.
- Rename file being downloaded by specifying "oldname:newname" for file name.
- Added FCE_SET_DATA_PORT to fceSetInteger.

Version 2.1: January 8, 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.
- Added CONN example program.

Version 2.2: October 3, 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 '\0' (in fceConnect) to skip USER and PASS processing.
- Performance improvements.
- Added MGET and PROXY example programs.

Version 2.3: November 15, 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.
- Added 200ms to minimum wait after sending password.
- Call control queue before each command.
- "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.
- Any keycode matches the shareware DLL.
- 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: April 19, 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).
- Added Microsoft VC.NET example.
- Added Microsoft C# example.

Version 2.5: July 20, 2005

- Added fceShortToByte and fceByteToShort functions.
- 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.c example program.

Version 2.6: January 17, 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 1, 2008

- 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 LoadLib and MDTM example programs.

Version 3.0: Sep 14, 2009

- Added support for 64 bit Windows (FCE64.DLL)
- Added fceIsConnected function
- Added fceToInteger function
- Added Visual Studio 2008 support

Version 3.1: July 6, 2011

- Log file is now time-stamped
- Added diagnostics to fceFileLength
- Function fceGetLocalFList no longer counts subdirectories
- Added fceGetFileSize function
- Added fceGetFileTime function
- Added MinGW project support
- Added Visual Studio 2010 support

Version 3.2: May 21, 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: April 22, 2014

- Fixed problem with connecting w/o blocking.
- Fixed problem with server name being corrupted in the FCE log file.
- Added project files for Visual C/C++ 2013.
- Added project files for Visual C/C++ 2012.
- Added support for Digital Mars C compiler.
- Added function fceGetSubDirs()
- Added FCE_SET_DEBUG_LEVEL to fceSetInteger().
- Added Embarcadero C++ Builder XE library files.

Version 3.4: October 28, 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.

Version 4.0: October 10, 2023

- Fixed problem: In some cases FCE functions returned 1 instead of -1
- Log file displays filename passed to fceGetFile and fcePutFile
- Data no longer written to log file
- Default write size buffer increased from 1024 to 4096
- Default is set to ASCII mode (as opposed to binary)
- Added "Not connected to server" error message.(FCE_NOT_CONNECTED)
- I/O buffer increased to 65536 bytes.