Client / Server Communications

for Visual Basic

Programmer's Manual

(CSC_4VB)

Version 7.1

January 16, 2018

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: www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **Client / Server Communications Library for Visual Basic (CSC4VB)** is a toolkit that allows software developers to quickly develop server and client TCP/IP and UDP applications in Visual Basic or Visual Basic .NET.

The **Client / Server Communications Library** (CSC) is a component library that uses the Windows API to create **client** and **server** programs (Win32 and Win64) that can communicate with each other across any TCP or UDP network such as the Internet or a private network (intranet or LAN [local area net]).

CSC can be used to communicate with other **CSC** programs, or they can be used to communicate with other TCP programs such as DNS, POP3, SMTP, FTP, HTTP, etc. CSC can also be used to connect to devices such as a relay device, scale device, GPS device or embedded computer device that is controlled by sending commands to its TCP IP address.

CSC4VB includes multiple Visual Basic example programs demonstrating client/server protocols, including examples that connect to HTTP (web) and POP3 servers as well as encrypt files. **CSC** can be used with our AES **Advanced Encryption Library** (**AES4VB**) if strong encryption is desired.. All examples compile using VB-4.0 through VB-6.0 or Visual Studio. Examples for Visual Basic for Applications (VBA) are also provided.

The **Client / Server Communications Library for Visual Basic (CSC4VB)** component library supports and has been tested with all versions of Microsoft Visual Basic (VB 4.0 – VB 6.0), Microsoft Visual Studio .NET Framework and Microsoft Visual Studio through Visual Studio 2013. **CSC4VB** can also be used with any VBA (Visual Basic for Applications) language such as Excel, Access, MS Office, etc. as well as with PowerBuilder.

Both Win32 DLL and Win64 DLLs are provided. **CSC4VB** runs under all versions of Windows (95/98/ME/2000/2003/2012/NT/XP/Vista/Windows7/Windows8). The **Client / Server Communications Library SDK** DLLs (CSC32.DLL and CSC64.DLL) can also be used from any language (Visual C++, .NET, Delphi, Visual FoxPro, COBOL, Xbase++, Visual dBase, etc.) capable of calling the Windows API.

The **Client/Server Communications Programmer's Manual** provides information needed to compile programs using CSC in a Visual Basic or VB. NET environment.

When comparing the **Client/Server Communications Library** against our competition, note that:

1. CSC4VB is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. WIN32 and Win64 DLLs are included.
3. CSC4VB does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
4. CSC is fully threadable.
5. The CSC functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **Client/Server Communications Library** for C/C++ (CSC4C), Delphi (CSC4D), Xbase++ (CSC4XB), dBASE (CSC4DB) and Visual FoxPro (CSC4FP). All versions of the **CSC** library use the same DLLs (CSC32.DLL and CSC64.DLL). However, the examples provided for each version are written for the specified programming language.

For the latest version of the CSC software, see

http://www.marshallsoft.com/client-server-communication.htm

Our goal is to provide a robust communication component library that you and your customers can depend upon. A fully functional 30-day evaluation version is available. Contact us if you have any questions.

1.1 Features

Some of the many features of the **Client/Server Communications Library** component are as follows:

- Supports both 32-bit and 64-bit Windows.
- Supports both UDP and TCP protocols.
- Can be used to create both **clients** and **servers**.
- Supports "one time" passwords for improved security.
- Can encrypt/decrypt data and files being transmitted.
- Use with the MarshallSoft **AES Encryption Library** for strong encryption.
- Supports challenge response authentication.
- Can send a Windows message when a connection is ready to accept.
- Can send a Windows message when incoming data is ready to be read.
- Can send and receive data buffers or entire files.
- Can connect to a device such as a relay device, scale device, GPS device or embedded computer device that is controlled by sending commands to its TCP IP address.
- Servers can handle multiple connections concurrently.
- Supports secure and private messaging
- Create chat server and clients
- Create client / server file transfer
- Create client programs to talk to TCP servers (POP3, IMAP, HTTP, SMTP, DNS)
- Create SMTP proxy programs extracting a copy of all recipient addresses
- Create POP3 proxy programs that filter incoming email for Spam
- Create HTTP proxy used to filter content
- Specify the maximum number of connections that the server will accept.
- Allows multiple servers and clients to run simultaneously.
- **Free** unlimited one-year email and phone tech support for registered users.
- License covers all programming languages.
- Royalty free distribution with a compiled application.
- Evaluation versions are fully functional. No unlock code is required.
- Is fully thread safe.
- Works with all versions of Microsoft Visual Basic (v4.0 through Visual Studio 2013)
- Works with Microsoft Visual Studio .NET
- Works with VBA (Visual Basic for Applications) such as Excel or Access as well as Power Builder.
- 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.
- Can be purchased with (or without) ANSI C source code.
- Updates are free for one year (Source code updates are separate).

A good selection of Visual Basic (and Visual Studio .Net) example programs with full source code is included. Refer to Section 6 for more details on each of the example programs.

```
auth_c          Authenticating client.
auth_s          Authenticating server.
Client          Simple client example program.
Controller      Sends command to IP address of device.
cscver          Displays CSC version and build
Download        Downloads file from web server.
FileGet         Receives a file.
FilePut         Sends a file.
FileGetX        Receives a file (extended protocol).
FilePutX        Sends a file (extended protocol).
FileSrv         File server example program.
FileCli         File client example program.
GetPrice        Downloads stock price from Yahoo.
Hello           Demonstrates use of CSC class.
RecvUDP         Receives UDP packet.
SendUDP         Sends UDP packet.
Server          Simple server example program.
uNetTime        UDP client gets Network Time.
```

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the first manual (CSC_4VB) in the set.

- CSC_4VB Programmer's Manual (CSC_4VB.PDF)
- CSC User's Manual (CSC_USR.PDF)
- CSC Reference Manual (CSC_REF.PDF)

The CSC_4VB Programmer's Manual (CSC_4VB) is the language specific (Visual Basic and Visual Studio .Net) manual. All language dependent programming issues are discussed in this manual. Information needed to compile programs in a VB environment is provided in this manual.

The CSC User's Manual (CSC_USR) discusses language independent issues. Information on Client / Server protocols as well as purchasing and license information is provided in the manual.

The CSC Reference Manual (CSC_REF) contains details on each individual CSC function.

The documentation is also provided on our web site at

http://www.marshallsoft.com/csc4vb.htm

1.3 Example Program

The following code segment attempts to connect to the server.

```
DataSock = cscClient(HostName, HostPort)
If DataSock < 0 Then
   Call DisplayError(Client, Code, "cscClient fails")
   Code = cscRelease()
   Exit Sub
End If
' wait for greeting message from server
Call DisplayText(Client, "Awaiting greeting message...")
' send WM_LBUTTONDOWN message when data is ready to be read
Code = cscDataMessage(Client.bReady.hWnd, DataSock, WM_LBUTTONDOWN)
```

Also see the example programs in the APPS sub-directory where CSC4VB was installed.

1.4 Installation

(1) Before installation of CSC4VB, a Visual Basic compiler should already be installed on your system and tested. Note that Visual Basic 4.0 or above is required in order to create Win32 programs.

(2) Unzip CSC4VB70.ZIP (evaluation version) or CSCxxxxx.ZIP (registered version; xxxxx is the Customer ID) using any Windows unzip program.

(3) Run the installation program SETUP.EXE which will install all CSC4VB files. SETUP will also copy CSC32.DLL and CSC64.DLL to the Windows directory. Note that no DLL registration is required.

1.5 Uninstalling

Uninstalling CSC4VB is very easy. First, delete the CSC4VB project directory created when installing CSC4VB. Second, delete CSC32.DLL and CSC64.DLL from your Windows directory, typically C:\WINDOWS for Windows 95/98/Me/XP/2003/Vista/Win7/Win8 or C:\WINNT for Windows NT/2000.

1.6 Pricing

A developer license for the Client/Server Communications Library can be purchased for $115 USD (or $195 USD with source code [ANSI C] to the library DLL). Purchasing details can be found in Section 1.4, "How to Purchase", of the CSC User's Manual (CSC_USR).

Also see INVOICE.TXT or http://www.marshallsoft.com/order.htm

1.7 Updates

When a developer license is purchased for CSC, the developer will be sent a registered DLL plus a license file (CSCxxxxx.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, the developer license must be updated to be able to download updates. The license can be updated for:

- $30 if the update is ordered within one year of the original purchase (or previous update).
- $55 is the update is ordered between 1 and 3 years of the original purchase (or previous update).
- $75 if the update is ordered after three years of the original purchase (or previous update).

The update price includes technical support for an additional year. Note that the registered DLLs, (CSC32.DLL and CSC64.DLL) never expire. If source code was previously purchased, updates to the source code can be purchased for $40 along with the license update.

1.8 Keycode (License Key)

CSC32.DLL and CSC64.DLL each have a keycode encoded within it. The keycode is a 9 or 10 digit decimal number (unless it is 0), and will be found in the file KEYCODE.BAS or KEYCODE.VB. The keycode for the evaluation version is 0. The developer will receive a new key code when registering. The KEYCODE is passed to **cscAttach**.

If an error message (value -74) is returned when calling **cscAttach**, it means that the keycode in the CSC application does not match the keycode in the DLL. After registering, it is best to remove the evaluation version of CSC32.DLL and CSC64.DLL from the Windows search.

2 Library Overview

The **Client/Server Communications Library for Visual Basic** has been tested on multiple computers running Windows XP through Windows 10.

The CSC4VB library has also been tested with several Visual Basic compilers, from VB 4.0 .0 through VB 6.0 .0 and Visual Studio .Net. CSC can also be used with Power Builder as well as 32-bit or 64-bit VBA applications such as Microsoft Office, EXCEL and ACCESS.

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 CSC4VB files are copied to the directory specified (default \CSC4VB). Three sub-directories are created, as follows:

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

2.1 Dynamic Link Libraries

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

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

The following files can be found in the DLL sub-directory when SETUP is run:

```
csc32.dll   - Win32 version of CSC
csc64.dll   - Win64 version of CSC
```

2.2 Win32 STDCALL and DECLSPEC

CSC32 is written in ANSI C and is compiled using the _stdcall and _declspec keywords. This means that CSC32 uses the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. Leading underscores and trailing "@size" strings are not added to function names.

The CSC32.DLL and CSC64.DLL functions may be called from any Windows application program capable of calling the Windows API provided that the proper declaration file is used.

2.3 Dynamic Strings

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

When passing a string buffer to a DLL function into which text will be copied, it is strongly recommended that the local string be allocated immediately before use. For example, a string buffer is passed to the user defined dllGetMessage function , which copies a text message into it. Note that `SPACE$(80)` is called immediately before dllGetMessage.

```
Dim Code As Integer
Dim Buffer As String * 80
' allocate buffer just before call to dllGetMessage
Buffer = SPACE$(80)
' copy message into 'Buffer'
Code = dllGetMessage(Buffer, 80)
' message text is now in 'Buffer'
```

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

2.4 CSCVB Class

The CSC class "cscClass" (cscClass.cls) is a Visual Basic class wrapper for making calls to CSC32.DLL and CSC64.DLL. The class name for each function is the same as the DLL function, except the leading "**csc**" is replaced by "**f**".

The functions that return strings do so by use of the "String Result" property. Instantiate **cscClass** as any other class in VB:

```
Dim X As New cscClass
```

Classes were added to Visual Basic beginning with version 5.0, and are required in order to compile **cscClass**

2.5 Visual Studio

There are a few differences between VB 4.0 /5/6 and Visual Studio that affect writing programs that use CSC.

(1) Variables that are declared “As Long” in VB 4/5/6 are declared “As Integer” in Visual Studio .Net.

(2) Fixed length strings are not supported in Visual Studio .Net. When calling any CSC function that can return a string, memory for the string variable must be allocated first. For example:

```
Buffer = Space(80)
Code = cscGetString(0, CSC_GET_REGISTRATION, Buffer, 80)
```

(3) Some VB functions must be fully qualified. For example, instead of `LEFT`, use `Microsoft.VisualBasic.Left`

(4) The module CSC32.VB (not CSC32.BAS) (or CSC64.VB) must be included in all Visual Studio .Net programs.

2.6 Visual Basic for Applications (VBA)

The **Client/Server Communication** component library can be used with Microsoft VBA applications such as EXCEL, ACCESS, and Microsoft Office.

Start EXCEL (or other 32-bit Office VBA program such as WORD or ACCESS), then enter design mode. Enable the "Controls Toolbox", choose "Tools" on the menu bar, then "Customize", and then check "Control Toolbox". From the control toolbox, choose and position a "Command Button". This will create code that looks like

```
Private Sub CommandButton1_Click()

End Sub
```

Replace the generated code with MODULE32.BAS. The easiest way to do this is to paste from the clipboard. Edit the 'HostName' in this code, using the name of the computer (or IP address in dotted decimal notation) where the server application (see SERVER.VBP) is running.

Exit design mode and then press the command button to start the CSC VBA example program.

2.7 PowerBuilder

CSC can also be used with 32 bit Power Builder applications. See PBUILDER.TXT in the \APPs subdirectory for more information.

`CSC32.PBI` : Power Builder declaration file.

2.8 Using Threads

CSC4VB is thread safe, and can be used from any Windows application capable of using threads.

The "AddressOf" operator, which was added to Visual Basic beginning in version 5.0, is required in order to create and run a Win32 thread in Visual Basic.

2.9 Adding CSC4VB to a Project

Copy CSC32.BAS (if running VB 4.0 /5/6), or CSC32.VB/CSC64.VB (if running Visual Studio .Net) into the same directory (folder) as the application program to which CSC code is to be added to. The files can be found in the APPS sub-directory (folder) created when SETUP was run, usually C:\CSC4VB\APPS.

2.9.1 Adding CSC4VB to a VB 4.0 .0, 5.0, or 6.0 Project

Open the existing project with "File", "Open Project". Then choose "Insert", "Module", then add CSC32.BAS and KEYCODE.BAS to the project. If prompted to add "DAO 2.50 Object Library", choose "no".

CSC functions can now be called from a VB program.

2.9.2 Adding CSC4VB to a Visual Studio .Net Project

Open the existing project with "File", "Open Project". Then choose "Project", "Add Module", then add CSC32.VB and KEYCODE.VB to the project. CSC functions can now be called from a Visual Studio .Net program.

2.10 Error Display

The error message text associated with CSC error codes can be displayed by calling **cscErrorText**.

```
Private Sub DisplayError(ByVal ErrCode As Integer)
Dim Code As Integer
Dim Buffer As String
Code =  cscErrorText(ErrCode, Buffer, 127)
If Code > 0 Then
  MsgBox( Microsoft.VisualBasic.Left(Buffer, Code) )
End If
End Sub
```

2.11 Example Client/Server Protocol

Several of the **Client/Server Communications Library** demonstration programs use the following example protocol:

(1) The server must be running first at a specified IP address using a specified port number known to both client and server. A host name may be used instead of an IP address. The server waits for a connection attempt by a client.

(2) The client attempts to connect to the server.

(3) The server accepts the connection from the client, and then sends its greeting message, such as:

"CSC Example Server"

(4) The client receives the server's greeting message.

(5) The client sends a request (command) string to the server.

(6) The server receives the client's request.

(7) The server sends back its response string.

(8) Repeat steps (5), (6), and (7) until done.

(9) The client closes its connection to the server.

The server responds with the following response strings when presented with the corresponding requests (REQ) from the client:

REQ	Response String	Request Example	Response Example
WHO	Sends name of the server.	WHO	W_SERVER
VER	Sends server version #.	VER	7.1
BYE	OK (then disconnects)	BYE	OK
ECH	Sends string after "ECHO "	ECHO Hello	Hello

The above protocol is just an example. The programmer can create whatever protocol is required. Request strings can be any length, although it is best to keep them as short as possible..

Also refer to PROTOCOL.TXT in the CSC4VB\DOCS subdirectory.

3 Compiler Issues

The **Client/Server Communication Library for Visual Basic** component library supports and has been tested with all versions of Microsoft Visual Basic including:

- Visual Basic 4, 5, 6
- Visual Basic .NET
- Visual Basic .NET 2003
- Visual Studio 2005
- Visual Studio 2008
- Visual Studio 2010
- Visual Studio 2012
- Visual Studio 2013
- Visual Studio 2015

CSC can also be used with any VBA language such as Excel, Access, MS Office as well as PowerBuilder.

3.1 Compiling CSC

The **Client/Server Communication** component library (CSC32.DLL) is written in standard ANSI C (CSC32.C), and has been compiled using Microsoft Visual C/C++ with the STDCALL and DECLSPEC compiler keywords. Source code for the CSC library is provided in the registered version (if ordered) only.

For more information on the C/C++ version of CSC, download the latest version of CSC4C from our web site at http://www.marshallsoft.com/csc4c.htm.

3.2 Compiling Example Programs

Visual Basic project files (.VBP) are provided for all example VB 4/5/6 programs.

Visual Basic NET (Visual Studio) project files (.VBPROJ) are provided for the Visual Studio .Net example programs.

3.3 Explicitly Loading a CSC DLL

When an application program runs that makes calls to CSC32.DLL or CSC64.DLL, the Windows operating system will locate CSC32.DLL (CSC642.DLL) by searching the directories as specified by the Windows search path. If the CSC32.DLL (CSC64.DLL) is placed in the \WINDOWS directory (or \WINNT for Windows NT/2000), it will always be found by Windows.

CSC32.DLL or CSC64.DLL can be loaded from an explicit location by replacing “CSC32.DLL” in CSC32.BAS or CSC32.VB by the full path. For example, to load CSC32.DLL from C:\CSC4VB\APPS, the first entry in would be:

```
Declare Function cscAcceptConnect Lib "C:\CSC4VB\APPS\CSC32.DLL" (ByVal vSock
As Long) As Long
```

For Win64, substitute CSC64 for CSC32 in the above.

4 Example Programs

The example programs are designed to demonstrate the various capabilities of CSC4VB. The best way to become familiar with CSC4VB is to study and run the example programs.

The example programs can be compiled with the version of Visual Basic indicated below. All example programs are located in the CSC4VB\APPS sub-directory.

4.1 CSCVER

The CSCVER ("CSC Version") example program displays the CSC version number. This is the first program to compile and build since it verifies that CSC DLL is installed properly. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	cscver.vb	
Visual Studio /VB.NET	cscver.vbproj	
Visual Studio 2008	cscver(VS2008).vbproj	cscver(VS2008)x64.vbproj
Visual Studio 2010	cscver(VS2010).vbproj	cscver(VS2010)x64.vbproj
Visual Studio 2012	cscver(VS2012).vbproj	cscver(VS2012)x64.vbproj
Visual Studio 2013	cscver(VS2013).vbproj	cscver(VS2013)x64.vbproj

4.2 SendUDP

SendUDP transmits a UDP data packet. Also see the RecvUDP example program.

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	SendUDP.vb	
Visual Studio /VB.NET	SendUDP.vbproj	
Visual Studio 2008	SendUDP(VS2008).vbproj	SendUDP(VS2008)x64.vbproj
Visual Studio 2010	SendUDP(VS2010).vbproj	SendUDP(VS2010)x64.vbproj
Visual Studio 2012	SendUDP(VS2012).vbproj	SendUDP(VS2012)x64.vbproj
Visual Studio 2013	SendUDP(VS2013).vbproj	SendUDP(VS2013)x64.vbproj

4.3 RecvUDP

RecvUDP receives UDP packets, including multicast UDP packets. Also see the SendUDP example program.

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	RecvUDP.vb	
Visual Studio /VB.NET	RecvUDP.vbproj	
Visual Studio 2008	RecvUDP(VS2008).vbproj	RecvUDP(VS2008)x64.vbproj
Visual Studio 2010	RecvUDP(VS2010).vbproj	RecvUDP(VS2010)x64.vbproj
Visual Studio 2012	RecvUDP(VS2012).vbproj	RecvUDP(VS2012)x64.vbproj
Visual Studio 2013	RecvUDP(VS2013).vbproj	RecvUDP(VS2013)x64.vbproj

4.4 Client

CLIENT is a VB example program that operates as a client that connects to the example server programs. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	Client.vb	
Visual Studio /VB.NET	Client.vbproj	
Visual Studio 2008	Client(VS2008).vbproj	Client(VS2008)x64.vbproj
Visual Studio 2010	Client(VS2010).vbproj	Client(VS2010)x64.vbproj
Visual Studio 2012	Client(VS2012).vbproj	Client(VS2012)x64.vbproj
Visual Studio 2013	Client(VS2013).vbproj	Client(VS2013)x64.vbproj

4.5 Server

SERVER is a VB example program that operates as a server that accepts connections from the example client program (CLIENT). SERVER accepts a maximum of 3 connections (clients) at any one time. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	Server.vb	
Visual Studio /VB.NET	Server.vbproj	
Visual Studio 2008	Server(VS2008).vbproj	Server(VS2008)x64.vbproj
Visual Studio 2010	Server(VS2010).vbproj	Server(VS2010)x64.vbproj
Visual Studio 2012	Server(VS2012).vbproj	Server(VS2012)x64.vbproj
Visual Studio 2013	Server(VS2013).vbproj	Server(VS2013)x64.vbproj

4.6 Auth_C

AUTH_C is a VB example program that operates as a client implementing authenticated (challenge/response) connections. Project files are:

Authen_c.vbp	Visual Basic 4/5/6

4.7 Auth_S

AUTH_S is a VB example program that operates as a server implementing authenticated (challenge/response) connections. Project files are:

Authen_s.vbp	Visual Basic 4/5/6

4.8 Controller

Controller is an example program that sends a byte command to a device such as

1. Relay Devices
2. Scale Devices
3. GPS Receivers
4. Embedded Computer Devices

that is controlled by sending commands to its TCP IP address.

In order to control a network device, you must know 3 things:

1. The name or IP address of the network device.
2. The port on which to contact the network device.
3. The protocol & command set to use with the network device.

Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	Controller.vb	
Visual Studio /VB.NET	Controller.vbproj	
Visual Studio 2008	Controller(VS2008).vbproj	Controller(VS2008)x64.vbproj
Visual Studio 2010	Controller(VS2010).vbproj	Controller(VS2010)x64.vbproj
Visual Studio 2012	Controller(VS2012).vbproj	Controller(VS2012)x64.vbproj
Visual Studio 2013	Controller(VS2013).vbproj	Controller(VS2013)x64.vbproj

4.9 Download

Download is an example client that connects to the MarshallSoft web site (HTTP server) and downloads a file from the `./files` directory. Project files are:

```
Download.vbp            Visual Basic 4/5/6
```

4.10 FileGet

FILEGET is a VB example program that operates as a server, and receives files from the FilePut client. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	FileGet.vb	
Visual Studio /VB.NET	FileGet.vbproj	
Visual Studio 2008	FileGet(VS2008).vbproj	FileGet(VS2008)x64.vbproj
Visual Studio 2010	FileGet(VS2010).vbproj	FileGet(VS2010)x64.vbproj
Visual Studio 2012	FileGet(VS2012).vbproj	FileGet(VS2012)x64.vbproj
Visual Studio 2013	FileGet(VS2013).vbproj	FileGet(VS2013)x64.vbproj

4.11 FileGetX

FILEGETX is a VB example program that operates as a server, and receives files from the FilePutX client. FileGetX uses the cscGetFileExt function (see section 2.10 in CSC Users Manual). Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	FileGetX.vb	
Visual Studio /VB.NET	FileGetX.vbproj	
Visual Studio 2008	FileGetX(VS2008).vbproj	FileGetX(VS2008)x64.vbproj
Visual Studio 2010	FileGetX(VS2010).vbproj	FileGetX(VS2010)x64.vbproj
Visual Studio 2012	FileGetX(VS2012).vbproj	FileGetX(VS2012)x64.vbproj
Visual Studio 2013	FileGetX(VS2013).vbproj	FileGetX(VS2013)x64.vbproj

4.12 FilePut

FILEPUT is a VB example program that operates as a client, and transmits files to the FileGet server. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	FilePut.vb	
Visual Studio /VB.NET	FilePut.vbproj	
Visual Studio 2008	FilePut(VS2008).vbproj	FilePut(VS2008)x64.vbproj
Visual Studio 2010	FilePut(VS2010).vbproj	FilePut(VS2010)x64.vbproj
Visual Studio 2012	FilePut(VS2012).vbproj	FilePut(VS2012)x64.vbproj
Visual Studio 2013	FilePut(VS2013).vbproj	FilePut(VS2013)x64.vbproj

4.13 FilePutX

FILEPUTX is a VB example program that operates as a client, and transmits files to the FilePutX server. FilePutX uses the cscPutFileExt function (see section 2.10 in CSC Users Manual). Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	FilePutX.vb	
Visual Studio /VB.NET	FilePutX.vbproj	
Visual Studio 2008	FilePutX(VS2008).vbproj	FilePutX(VS2008)x64.vbproj
Visual Studio 2010	FilePutX(VS2010).vbproj	FilePutX(VS2010)x64.vbproj
Visual Studio 2012	FilePutX(VS2012).vbproj	FilePutX(VS2012)x64.vbproj
Visual Studio 2013	FilePutX(VS2013).vbproj	FilePutX(VS2013)x64.vbproj

4.14 **FileCli** ("File Client")

The FileCli example client program connects to the FileSrv example server program in order to upload or download files. See the comments in FileCli.frm or see PROTOCOL.TXT for a description of the protocol used. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	FileCli.vb	
Visual Studio /VB.NET	FileCli.vbproj	
Visual Studio 2008	FileCli(VS2008).vbproj	FileCli(VS2008)x64.vbproj
Visual Studio 2010	FileCli(VS2010).vbproj	FileCli(VS2010)x64.vbproj
Visual Studio 2012	FileCli(VS2012).vbproj	FileCli(VS2012)x64.vbproj
Visual Studio 2013	FileCli(VS2013).vbproj	FileCli(VS2013)x64.vbproj

4.15 **FileSrv** ("File Server")

The FileSrv example server program can accept multiple connections and allows clients to upload or download files. See the comments in FileSrv.frm or see PROTOCOL.TXT for a description of the protocol used. Project files are:

VB Compiler	32-bit	64-bit
Visual Basic 4/5/6	FileSrv.vb	
Visual Studio /VB.NET	FileSrv.vbproj	
Visual Studio 2008	FileSrv(VS2008).vbproj	FileSrv(VS2008)x64.vbproj
Visual Studio 2010	FileSrv(VS2010).vbproj	FileSrv(VS2010)x64.vbproj
Visual Studio 2012	FileSrv(VS2012).vbproj	FileSrv(VS2012)x64.vbproj
Visual Studio 2013	FileSrv(VS2013).vbproj	FileSrv(VS2013)x64.vbproj

4.16 Hello

HELLO is an example program that demonstrates the use of the CSC class `cscClass.cls`.. Project files are:

Hello.vbp	Visual Basic 5/6 (not VB4)

4.17 uNetTime

uNetTime is an example UDP client that connects to a Network Time Server (on well known port 37) and gets the network time (seconds since 1 January 1900 GMT) from the server. The default server is

```
time-A.timefreq.bldrdoc.gov
```

Project files are:

GetPrice.vbp	Visual Basic 4/5/6

5 Revision History

Version 2.0: October 15, 2004

Visual Basic version of CSC initial release.

Version 3.0: September 9, 2005.

- Corrected problem not freeing socket when connect fails.
- Increased file length from an 8-digit number to a 9-digit number in cscGetFile and cscPutFile.
- Added CSC_SET_DEBUG_LEVEL to allow for multiple debug levels.
- Increased the default file packet length from 4K to 8K.
- Added cscIsConnected function to allow a server to check if a client is still connected.
- Added cscShortToByte and cscByteToShort.
- Added CSC_GET_BUFFER_SIZE - returns buffer size used in cscGetFile and cscPutFile.
- Added ability to cancel file transfer from remote side.
- Added FileSrv and FileCli example programs.

Version 4.0: April 11, 2007

- Added CSC_GET_DAYS_LEFT to cscGetInteger
- Added CSC_FILE_TOO_LARGE error code.
- Allow multiple listen sockets.
- Changed: cscAttach, cscAwaitConnect, cscAcceptConnect, cscServer
- Removed: cscSendMessage
- Added: cscDataMessage, cscConnectMessage
- Increased NBR_DATA_SOCKS from 128 to 1024.
- Modified CSC_SET_FILE_PATH to also set path for individual data socket.
- Append "-1", "-2", etc to filename if file already exists for cscGetFile
- Added VS_SET_LINGER to cscSetInteger
- Default linger time reduced to 200 ms
- cscAttach now returns DaysLeft (for evaluation version)

Version 5.0: October 2, 2008.

- cscSetInteger(Sock, CSC_SET_LINGER, LingerTime) returns LingerTime
- Added cscLaunch()
- Increased NBR_LISTEN_SOCKS to 64
- Added GetPrice example program.
- Added POP3Stat example program.
- Added append mode (CSC_SET_FILE_APPEND).
- Added overwrite mode (CSC_SET_FILE_OVERWRITE).
- Added cscCryptoGetFile and cscCryptoPutFile
- Added cscCryptoGetData and cscCryptoPutData
- Added cscFillRandom
- Added CSC_DATA_SIZE error code (BufLen too big)
- Added CSC_SET_DELAY_TIME
- Increased TCP buffer size to 1500
- Socket portion rewritten for significantly faster speed.

Version 6.0: July 24, 2009

- Supports 64-bits (CSC64.DLL).
- Added cscPutPacket and cscGetPacket.
- Added cscCryptoPutPacket and cscCryptoGetPacket.
- Added CSC_SET_MAX_PACKET_SIZE and CSC_GET_MAX_PACKET_SIZE.
- Changed: NBR_DATA_SOCKS to 1000, NBR_LISTEN_SOCKS to 50.
- Changed: MAX_FILE_BUFFER_SIZE to 30000.
- Changed: DEFAULT_FILE_BUFFER_SIZE to 10000.
- Added CSC_SET_PAD_TX_INDEX and CSC_SET_PAD_RX_INDEX.
- Added cscDataCRC and cscFileCRC.
- cscGetInteger(Chan, CSC_GET_SOCKET) returns actual socket.
- Added cscCreateUDP, cscGetUDP, cscCreateUDP.
- Added cscNetToHost32.

Version 6.1: September 1, 2010

- Fixed: cscCreateUDP not saving socket so not closed later.
- Changed: default connect timeout from 60 seconds to 10 seconds.
- Changed: Pass RotateCount < 0 to cscResponse to return rightmost 31 bits of encrypted binary value.
- Fixed: vSock slot freed when connect fails.
- Added: cscReadSize() returns # bytes ready to be read.
- Added: support for MinGW C compiler (gcc).
- Added: support for Microsoft VS 2010.
- Added: Controller example program.

Version 6.2: February 21, 2012

- Added function cscMakeDotted4()
- Corrected Julian date function
- Added additional diagnostics (to detect congestion) in csc-vs.c
- Fixed cscReadSize(), worked only for vSock 0.
- Fixed CSC_SET_TIMEOUT_VALUE not being passed to csc-vs
- Fixed cscChallenge uses leading zeros to pad to 8 chars
- Changed DEFAULT_PACKET_TIMEOUT to 35000
- Added error text for VS_* errors.
- Added functions cscPutFileExt & cscCryptoPutFileExt
- Added functions cscGetFileExt & cscCryptoGetFileExt
- Fixed problem receiving file with same name from two clients

Version 6.3: July 3, 2013

- Fixed: cscCryptoGetFileExt and cscCryptoPutFileExt were not using local file path.
- Added: cscFillRandom(*,*,Seed) uses random seed if passed seed is zero.
- Added: cscSetInteger(Port, CSC_SET_CLOSE_TIMEOUT, Tics) sets max tics before socket is forced closed.
- Added: New example programs that use AES (Advanced Encryption Standard).
- Fixed: cscResolve was sometimes returning bogus IP addresses after the first.

Version 7.0: March 27, 2015

- Added cscMulticast() that receives multicast UDP packets.
- Added cscClientExt() that binds to a local IP address (for multi-homed computers)
- Added Microsoft VS2012 and VS2013 project files.
- Fixed bogus CSC_BAD_OFFSET error sometimes returned by crcCryptoPutFile().

Version 7.1: January 16, 2018

- Fixed problem with cscFileCRC()
- Fixed problem with cscCryptoPutFileExt() - data was being incorrectly appended.
- Added more internal diagnostics.
- Added cscTestDotted() - returns TRUE if dotted address in correctly formatted.
- Fixed problem in cscRelease() - log file was being closed prematurely.
- Increased MAX_DATA_SIZE from 30000 to 50000 bytes.
- Added CSC_SET_SOCK_REUSE [to cscSetInteger], enabling an app to close the listening socket and immediately reopen without error.