

Client / Server
Communications
Programmer's Manual

(CSC_4C)

Version 6.1.1

August 29, 2010

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

Copyright (C) 2010
All rights reserved

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

web: www.marshallsoft.com

MARSHALLSOFT is a registered trademark of MarshallSoft Computing.

TABLE OF CONTENTS

1	Introduction	Page 3
1.1	Features	Page 4
1.2	Documentation Set	Page 6
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
2	CSC Library Overview	Page 8
2.1	Dynamic Link Library	Page 8
2.2	Keycode	Page 8
2.3	Win32 STDCALL and DECLSPEC	Page 8
2.4	Console Mode	Page 9
2.5	Example Client / Server Protocol	Page 9
2.6	Static Linking	Page 10
2.7	Calling CSC from C++	Page 10
2.8	Adding CSC To An Existing Program	Page 10
2.9	Error Display	Page 11
2.10	Explicitly Loading a CSC DLL	Page 11
2.11	Targeting a 64-Bit CPU	Page 12
3	Compiler Issues	Page 13
3.1	Compiling Using an IDE	Page 13
3.2	Command Line Tool Setup	Page 14
3.3	Command Line Batch Files	Page 15
3.4	Command Line Makefiles	Page 15
4	Supported Compilers	Page 16
4.1	Microsoft Visual C++	Page 16
4.2	Microsoft Visual Studio C++	Page 18
4.3	Microsoft Visual Studio 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/C++	Page 22
5	Compiling Programs	Page 23
5.1	Compiling CSC Source Code	Page 23
5.1	Static Libraries	Page 23
5.2	Compiling Example Programs	Page 24
6	Example Programs	Page 26
7	Revision History	Page 31

1 Introduction

The **Client / Server Communications Library for C/C++ (CSC4C)** is a toolkit that allows software developers to quickly develop server and client TCP/IP and UDP applications using C/C++ or .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 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.

CSC4C includes multiple C/C++ example programs demonstrating client/server protocols, including examples that connect to HTTP (web) and POP3 servers as well as encrypt files.

The **Client / Server Communications Library for C/C++ (CSC4C)** 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++, MinGW C++ and LCC-Win32 C compilers. **CSC4C** can also be used with most other C/C++ Windows compilers.

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

CSC4C runs under all versions of Windows (Windows 95, Windows 98, Windows ME, Windows 2000, Windows 2003, Windows NT, Windows XP, Windows 7, Vista and x64). The **Client / Server Communications Library SDK DLLs** (CSC32.DLL and CSC64.DLL) can also be used from any language (Visual Basic, VB.NET, Codegear Delphi, Visual FoxPro, COBOL, Xbase++, dBase, Microsoft Office, etc.) capable of calling the Windows API. Win32 and Win64 DLLs are provided.

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

1. **CSC4C** is a standard Windows DLL (NOT an OCX or ActiveX control) and is much smaller than a comparable OCX or ActiveX control.
2. **CSC4C** does NOT depend on ActiveX or Microsoft Foundation Class (MFC) libraries or similar "support" libraries.
3. **CSC** is fully threadable.
4. The **CSC** functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **Client/Server Communications Library** for Visual Basic (CSC4VB), 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 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 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.
- 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.
- Can specify the maximum number of connections that the server will accept.
- Allows multiple servers and clients to run simultaneously.
- Create chat server and clients.
- Create client / server file transfer.
- Create client programs to talk to TCP servers (POP3, IMAP, HTTP, FTP, SMTP, DNS, etc.).
- 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.

- **Free** technical support and updates for one year.
- License covers all programming languages.
- Royalty free distribution with a compiled application.
- Evaluation versions are fully functional. (60 day). No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Supports Windows 95/98/Me/NT/2000/2003/XP/Vista/Window 7.
- Implemented as a **standard** Windows DLL, which will work with all versions of Windows.
- Both Win32 and Win64 DLLs are included.
- 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/C++ (v4.0 through Visual Studio 2010).
- Works with Microsoft Visual Studio .NET and Visual C#.
- Works with Borland C/C++ (v5.0, v5.5, and Borland C++ Builder [all versions]).
- Also works with Microsoft Foundation Class, Watcom v11, MinGW and LCC-WIN32.
- 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 or Excel.
- Can be purchased with (or without) source code.
- Updates are free for one year (Source code updates are separate).
- Unlimited one-year email and phone tech support.
- Documentation online as well as in printable format.

A good selection of C/C++ (also Visual C++ .NET and C# .NET) example programs with full source code is included. Refer to Section 6 for more details on each of the example programs.

Legend:

Con = Console mode.
GUI = GUI mode.
VC = VC 4 thru 6.
VC/B/L = For VC 4 thru 6, Borland 5.0 thru 5.5, LCC
VCN = Visual C++ .NET
C# = Visual C# .NET (C Sharp)

auth_c.c	Authenticating client program (Con: VC/B/L).
auth_s.c	Authenticating server program (Con: VC/B/L).
bcj_prj.cpp	Displays CSC version, build, & registrations (Borland C++ Builder)
client.c	Client program (Con: VC/B/L).
control.c	Controls device by sending commands to its TCP IP address (Con: VC).
cscver.c	Displays CSC version, build, & registrations (Con: VC/B/L).
download.c	Downloads a file from HTTP (web) sever (Con: VC/B/L)
FileCli.c	File client connects to FileSrv to download files. (Con: VC/B/L).
FileGet.c	Received files - crypto example (Con: VC/B/L).
FilePut.c	Transmits files - crypto example (Con: VC/B/L).
FileSrv.c	File server handles concurrent connections (Con: VC/B/L).
GetPrice.c	Downloads stock price from finance.yahoo.com (Con: VC/B/L).
hello.cpp	Uses C++ class wrapper to display version, build, etc. (VC).
HellpCS.cs	Displays CSC version, build, and registrations screen (C#)
List.c	Lists all TCP/IP addresses (Con: VC/B/L).
POP3Stat.c	Returns the number of emails waiting on POP3 server (Con: VC/B/L).
Proxy.c	POP3 proxy program (Con: VC/B/L)
server.c	Server program (Con: VC/B/L)
server2.c	Server program. Handles concurrent connections (Con: VC/B/L).
uEcho_c.c	Example UDP echo client (Con: VC/B/L).
uEcho_s.c	Example UDP echo server (Con: VC/B/L).
uNetTime.c	Gets network time from UDP Network Time Server (Con: VC/B/L).
vc_client.cpp	Client program (Con: VCN).
vc_server.cpp	Server program. Handles concurrent connections (Con: VCN).
vc_vers.cpp	Displays CSC version, build, and registrations screen (VCN).
vc_w_client.cpp	Client program (GUI: VCN).
vc_w_server.cpp	Server program. Handles concurrent connections (GUI: VCN).
vers.c	Displays CSC version, build, & registrations (GUI: VC/B/L).
w_client.c	Client program (GUI: VC/B/L).
w_server.c	Server program. Handles concurrent connections (GUI: VC/B/L).

1.2 Documentation Set

The complete set of documentation is provided in Adobe PDF format. This is the first manual (CSC_4C.PDF) in the set.

- [CSC 4C Programmer's Manual](#) (CSC_4C.PDF)
- [CSC User's Manual](#) (CSC_USR.PDF)
- [CSC Reference Manual](#) (CSC_REF.PDF)

The CSC_4C Programmer's Manual is the language specific (C/C++) manual and provides information needed to install and compile example programs in a C/C++ environment.

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

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

Documentation is also provided online at <http://www.marshallsoft.com/csc4c.htm>

1.3 Example Program

The following example demonstrates some of the library functions.

```
#include "csc.h"
#include "keycode.h"

static char *HostName;
static char Temp[256];

void main(int argc, char *argv[])
{int Code;
 int Version;
 int Build;
 // attach (one data socket & no listen socket)
 Code = cscAttach(1, 0, (long)CSC_KEY_CODE);
 if(Code<0)
 {printf("ERROR %d: cannot attach\n", Code);
  exit(1);
 }
 // define diagnostics log file
 cscSetString(-1, CSC_SET_LOG_FILE, (char *)"cscver.log");
 // display version and build
 Version = cscGetInteger(-1, CSC_GET_VERSION);
 Build = cscGetInteger(-1, CSC_GET_BUILD);
 printf("CSC Version: %ld.%ld.%ld Build %d\n",
        0x0f&(Version>>8),0x0f&(Version>>4),0x0f&Version,Build);
 Code = cscGetString(-1, CSC_GET_REGISTRATION, (char *)Temp, 51);
 if(Code<0) printf("ERROR %d: Cannot get registration string\n", Code);
 else printf("Registration: %s\n", Temp);
 cscRelease();
}
```

1.4 Installation

(1) Before installation of CSC4C, a Windows C/C++ compiler should already be installed and tested. In particular, include command line tools when installing a compiler to be able to compile using command line makefiles. Refer to the file, MAKEFILE.TXT, for help with makefiles.

(2) Unzip CSC4C611.ZIP(evaluation version) or CSCxxxx.ZIP (registered version; xxxxx is the Customer ID).

(3) Run the installation program SETUP.EXE that will install all CSC4C files. SETUP will also copy CSC32.DLL and CSC64.DLL to the Windows directory. Note that no DLL registration is required.

All recent WIN32 C/C++ compilers support the "declspec" keyword. Microsoft Visual C++ (version 4.0 and up), Borland (version 5.0 and up), and LCC-Win32 compilers support the "declspec" keyword. Makefiles for Watcom (version 11.0 and up) are available on request.

1.5 Uninstalling

Uninstalling CSC4C is very easy. First, delete the CSC project directory created when installing CSC4C. Second, delete CSC32.DLL and CSC64.DLL from the Windows directory, typically C:\WINDOWS for Windows 95/98/Me/XP/2003/Vista/Win7 and C:\WINNT for Windows NT/2000.

1.6 Pricing

A developer license for the Client/Server Communications Library can be purchased for \$115 (or \$195 with source code to the library DLL). Purchasing details can be found in Section _1.4, "How to Purchase", of the CSC User's Manual (CSC_USR). (http://www.marshallsoft.com/csc_usr.pdf)

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 set of registered DLLs plus a license file (CSCxxxx.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 if the update is ordered between one and three 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.

2 CSC Library Overview

The **Client/Server Communications Library for C/C++** has been tested on multiple computers running Windows 95/98/Me/2003/XP/Vista/Vista/Windows 7/x64 and Windows NT/2000.

The CSC4C 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 C++ and LCC-Win32 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 CSC4C files are copied to the directory specified (default \CSC4C). Three sub-directories are created, as follows:

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

2.1 Dynamic Link Library

The **Client/Server Communications Library** (CSC) is implemented as 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 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".

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
```

Note: The LIB files for the above DLLs can be found the in the APPS directory

2.2 Keycode

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.H. The keycode for the evaluation version is 0. The developer will receive a new key code after 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.

2.3 Win32 STDCALL and DECLSPEC

CSC32 is compiled using the `_stdcall` and `_declspec` keywords. This means that CSC4C uses 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++ users can look at the CSC32 function names using the dumpbin.exe executable:
dumpbin /exports csc32.dll

2.4 Console Mode

CSC functions can be called from Win32/Win64 console mode programs. A "console mode" program is a Windows 95/98/NT/2000/Me/XP/Vista/Win7 WIN32/Win64 command line program running in a command window. Although console mode programs look like DOS programs, they are WIN32/Win64 programs that have access to the Win32/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.

2.5 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:

<u>REQ</u>	<u>Response String</u>	<u>Request Example</u>	<u>Response Example</u>
WHO	Sends name of the server.	WHO	W_SERVER
VER	Sends server version #.	VER	2.0
BYE	OK (then disconnects)	BYE	OK
ECH	Sends string after "ECHO "	ECH 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 \APPS subdirectory.

2.6 Static Linking

The registered user can also statically link CSC32.OBJ (or CSC64.OBJ) with their application, rather than making calls to the DLL's.

For example, to create an application that links CSC32.OBJ statically:

- (1) All application code that includes CSC.H must define `STATIC_LIBRARY` before including CSC.H
- (2) The application must link with WSOCK32 (for WIN32).

If using Microsoft Developer Studio, make these changes:

- (1) To the project file: Do not add CSC32.LIB to the 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 `csc32.obj` to "object/library modules:"
- (3) Add `#include "csc.h"` to all source files that make calls to CSC functions.

Alternatively, CSC32.C can be edited so that it can be compiled and linked like any other program file. In order to do this, remove all code from CSC32.C (provided when the source code is purchased) from

```
#ifndef STATIC_LIBRARY
```

through the following

```
#endif
```

2.7 Calling CSC from C++

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

CSC functions can also be called using the C++ class wrapper `fcsc`. Refer to HELLO.CPP for an example.

2.8 Adding CSC to an Existing Program

In order to call CSC functions from an existing program,

- (1) add

```
#include "csc.h"
```

to the application source code,

- (2) link with CSC32.LIB (for MSVC), CSC32BCB.LIB (Borland C/C++ and C++ Builder), CSC32.LIB (Watcom), or CSC32LCC (Win32/LCC), and recompile from source.

For Win64, link with CSC64.LIB rather than CSC32.LIB

2.9 Error Display

The error message text associated with **CSC** error codes can be displayed by calling **cscErrorText**. For example, if 'ErrCode' is returned by a CSC function and it is negative (indicating an error), the text associated with the error code can be found by calling **cscErrorText**, as demonstrated by the following code segment:

```
void DisplayError(int ErrCode, char *MsgPtr)
{int Len;
 char ErrBuff[128];
 printf("ERROR %d: ", ErrCode);
 if(MsgPtr) printf("%s: ", MsgPtr);
 Len = cscErrorText(ErrCode, (char *)ErrBuff, 127);
 if(Len>0) printf("%s\n", ErrBuff);
 else printf("\n");
}
```

Error codes and associated text are also written to the CSC log file, if it has previously been enabled by calling

```
cscSetString (Sock, CSC_SET_LOG_FILE, (char *)log-file-name)
```

2.10 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** (or **CSC64.DLL**) by searching the directories as specified by the Windows search path. If the CSC DLL is placed in the **\WINDOWS** directory (or **\WINNT** for Windows NT/2000), it will always be found by Windows.

However, **CSC32.DLL** (or **CSC64.DLL**) can also be loaded from a specified directory by using the **GetProcAddress Win32 API** function. For an example, refer to the **LoadLib.c** program.

2.11 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 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 CSC32.DLL. The 32-bit application code generated will be executed by the Windows WOW64 (Windows on Windows 64-bit) component.

2.11.1 Visual Studio C/C++: Versions 2005 through 2010

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).

2.11.2 Visual Studio Visual Basic: Versions 2005 through 2010

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".

2.12 64-bit CSC

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 CSC64.LIB instead of CSC32.LIB.

However, if a compiler generates 32-bit code, the application must be linked with CSC32.LIB even though it may be running on a 64-bit machine.

There are numerous CSC4C 64-bit example programs. 64-bit Visual Studio 2008 and 2010 project files include the following in the /APPS folder:

```
vc_FileGet(VS2008)x64.vcproj      vc_FilePut(VS2008)x64.vcproj
vc_server(VS2008)x64.vcproj      vc_client(VS2008)x64.vcproj
vc_w_client(VS2008)x64.vcproj    vc_w_server(VS2008)x64.vcproj
vc_vers(VS2008)x64.vcproj

vc_FileGet(VS2010)x64.vcxproj    vc_FilePut(VS2010)x64.vcxproj
vc_server(VS2010)x64.vcxproj    vc_client(VS2010)x64.vcxproj
vc_w_client(VS2010)x64.vcxproj  vc_w_server(VS2010)x64.vcxproj
vc_vers(VS2010)x64.vcxproj
```

The remaining 64-bit examples are contained in the archive (vc2008).x64.vcproj.zip. and (vc2010).x64.vcproj.zip.

3 Compiler Issues

CSC 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 the compiler's IDE. For Visual C/C++, "project makefiles" are used since they can be used by all versions of Microsoft Visual C++. When opening the workspace, select "makefiles(.mak)" for the file type.

Alternatively, for Visual C++ 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 CSCVER example program files hierarchy in the IDE (for 32-bit) should look like:

```
CSCVER.EXE
+++ CSCVER.C
+++ CSC32.LIB
```

Replace CSC32.LIB with CSC32BCB.LIB if using Borland C++ Builder, and with CSC32LCC.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 a 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 a compiler to the next.

In order to compile from the command line, command line compiler tools must be set up properly. Note that an option to install command line tools (or not to install them) is provided during installation of a C/C++ compiler. Refer to the compiler manufacturer's manual for details.

If necessary, the size of the environment table space can be increased by adding

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

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

For all compilers, the path should point to the compiler's BIN directory. For example, to add "C:\BC50\BIN" to an 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 the 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=
```

3.2.3 Watcom

Set the WATCOM environment variables to point to the 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
```

Watcom makefiles are available on request.

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 the 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 the 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.

See CSCVER\$.BAT for an example of a command line batch file for LCC-Win32. Similarly, command line batch files can be created for all of the example programs.

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 created when SETUP was run

borland50.zip	Borland C/C++ 5.0 makefiles.
borland55.zip	Borland C/C++ 5.5 makefiles.
c-builder.zip	Borland C++ Builder project makefiles.
lcc-win32.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.

4.0 Supported Compilers

The **Client/Server Communications Library for C/C++** has been tested with Microsoft Visual C++ (all versions including Visual Studio C++ and Visual Studio C# through Visual Studio 2010), Borland C/C++, Borland C++ Builder, Borland Turbo C/C++, Watcom C/C++, MinGW C++ and Lcc-Win32. 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++ programs can be compiled from either the command line or from within the Microsoft 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. CSC 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 SERVER._M_
```

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

The file MICROSOFT60.ZIP contains the Microsoft Visual 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 CSCVER.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 the 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 CSC32.LIB. Lastly, select "Build", then "Rebuild All". Be sure to specify /YX rather than /Yu in the 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 CSCVER.MAK.

To create a new project in MSVC v5.0, choose "File", then "New", then "Win32 Application" or "Win32 Console Application" and the project name. Check "Create new workspace". Select "Project", then "Add to Project". Add all filenames including any resource file (.RC) and CSC32.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 the 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, 2005, 2008 and 2010 (VC++ 7.0, VC++ 8.0, VC++ 9.0 and VC++ 10.0)

Open the VC project file (files ending in ".vcxproj", ".vcproj" or ".dsp"), build and run, as in previous version of Visual Studio. Visual Studio 2008 specific project files end with ".(VS2008).vcproj" for 32-bit applications and ".(VS2008)x64.vcxproj" for 64-bit applications. Visual Studio 2010 specific project files end with ".(VS2010).vcxproj" for 32-bit applications and ".(VS2010)x64.vcxproj" for 64-bit applications.

Also see section 4.2 below.

4.1.6 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++ and Visual 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:\CSC4C\APPS).

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

(1) Add

```
#include "csc.h"  
#include "keycode.h"
```

after the existing #include statements.

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

(3) Select "Configuration Properties", "Linker", "Input", "Additional Dependencies", and then type in "CSC32.LIB" into the edit box. Note that CSC32.LIB must be in the project directory along with all of the source files.

(4) Rebuild.

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

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

must be the first include statement in a program.

4.3 Microsoft Visual C#

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

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

```
cs_vers.csproj
ClientCS.csproj
```

In order to open an existing C# project, choose "File", "Open", and then "Project" from the Microsoft C# Development Environment. Specify the directory containing the C# project files (for example, C:\CSC4C\APPS).

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

- (1) Add

```
using System.Runtime.InteropServices;
```

to the application source code.

- (2) Add the contents of file `csc_funs.cs` to the application source code after

```
public class csc : System.Windows.Forms.Form
```

- (3) Add the constants from `csc_cons.cs` to the program as they are needed.

Look at the `cs_vers` and `ClientCS` example projects in the APPS directory.

- (4) Set "unsafe" compilation for any functions that calls CSC functions. For example

```
private unsafe void button1_Click(object sender, EventArgs e)
```

- (5) Verify that `AllowUnsafeBlocks` is set to true in the project file (`.vcproj`). That is,

```
AllowUnsafeBlocks = "true"
```

Before writing C# programs using CSC, look at the `HelloCS` and `ClientCS` example projects.

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 CSC32BCB.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 CSCVER._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 CSC32BCB.LIB to the project. CSC32BCB.LIB can also be created from CSC32.DLL using the Borland IMPLIB utility:

```
IMPLIB CSC32BCB.LIB CSC32.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, unresolved external references to the library functions is received in which each function name is in all upper case, then case sensitivity has NOT be turned off as outlined above.

NOTE2: If errors are received while 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 CSC32BCB.LIB is the LIB file used with Borland C++ Builder. CSC32BCB .LIB can be created from CSC32.DLL by using the Borland IMPLIB program:

```
IMPLIB CSC32BCB.LIB CSC32.DLL
```

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

4.7 Watcom C/C++

CSC4C 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 CSCVER._W_
```

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

```
CSCVER
```

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/C++

LCC-Win32 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://ww.cs.virginia.edu/~lcc-win32/>

To use our DLL's with LCC-Win32, link with CSC32LCC.LIB. This file can also be re-created using the LCC-Win32 utility BUILDLIB.

```
buildlib csc32.lcc csc32lcc.lib
```

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

```
lcc -DWIN32 csver.c  
lclnk csver.obj csc32.lib -subsystem:console
```

To compile the GUI mode example VERS,

```
lcc -DWIN32 vers.c  
lcc -DWIN32 paint.c  
lrc vers.rc  
lclnk vers.obj paint.obj csc32lcc.lib vers.res -subsystem:windows
```

Command files are used for LCC-Win32 rather than makefiles since the make utility that comes with LCC-Win32 does not work well (unlike the actual compiler).

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

4.9 MinGW C/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 server.c csc32.lib -o server.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.

5 Compiling Programs

At the current time (August 2010), Microsoft Visual Studio is the only C/C++ compiler that can generate 64-bit application code.

5.1 Compiling CSC Source Code

This section applies only to those who have purchased source code for the **Client/Server Communication Library**.

CSC32.DLL and CSC64.DLL have been compiled using Microsoft Visual C++, and are callable from applications written using Microsoft, Borland, or Watcom compilers. If CSC.C is recompiled using Borland or Watcom compilers, then the resulting CSC32.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 CSC32.DLL and CSC32.OBJ (for static linking).

In order to create CSC32.DLL, type:

```
NMAKE -f CSC32._M_
```

In order to create CSC32.OBJ (for static linking) type

```
NMAKE -f CSC32S._M_
```

Alternatively, CSC32.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`.

CSC64.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).

Refer also to 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 CSCVER._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 ".VCXPROJ" or ".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 CSCVER._M_).

5.2.3 Compiling Using Visual C#

CSC 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 CSCVER._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 For example (MAKE -f CSCVER._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 are provided. For example (e.g.: WMAKE -f CSCVER._W_)

5.2.8 Compiling Using LCC-WIN32

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

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

6 Example Programs

Many of the example programs are written in Win32 console mode. This was done in order to provide the clearest possible code, without the complication and complexity of GUI code. All console mode programs can be converted to GUI mode by adding the necessary Windows interface code. Example programs are located in the APPS sub-directory. Refer to Section 4.0 for help with compiling and using makefiles.

See section 1.1 "Features" above for a list of all example programs in a tabular format.

Project files are classified as follows:

- *.m_ Microsoft C/C++ makefile (command line).
- *.b_ Borland C/C++ 5.0 makefile (command line).
- *.i_ Borland C/C++ 5.5 makefile (command line).
- *.w_ Watcom C/C++ makefile (command line).
- *.dsp Microsoft v6.0 project file.
- *.vcproj Microsoft Visual C++ .NET project file.
- *.vcxproj Microsoft Visual C++ .NET project file (VS2010).
- *.csproj Microsoft Visual C# project file.

Microsoft Visual C/C++ Developer Studio files end in ".MAK" (or ".DSP" for v6.0) and can be loaded with "Open Workspace".

Other files ending with .MAK (and .BPR) include:

- BCB_PRJ.MAK Borland C++ Builder makefile (BCB version 1 through 3).
- BCB_PRJ.BPR Borland C++ Builder makefile (BCB version 4 and above).

Please refer to the CSC User's Manual (CSC_USR.DOC or CSC_USR.HTM) for basic information on Server / Client protocols.

Note: Contact us at info@marshallsoft with Subject "CSC4C HELP" if you are having problems running the evaluation version with Windows Server.

6.1 CSCVER

The first example program is the console mode program CSCVER (CSC Version) that displays the CSC library version number.

See the COMPILE comment in CSCVER.C for information on compiling.

6.2 VERS

VERS is a Windows (GUI mode) example C/C++ program that displays the CSC32.DLL Version, Build #, and registration string.

See the COMPILE comment in VERS.C for information on compiling.

6.3 VC_VERS

VC_VERS is a Visual C++ .Net Windows (GUI mode) example program that displays the CSC32.DLL Version, Build #, and registration string.

See the COMPILE comment in VC_VERS.CPP for information on compiling.

6.4 HELLO

HELLO is a console mode example C++ program that displays the CSC32.DLL Version, Build #, and registration string using the CSC class (fcsc.cpp).

See the COMPILE comment in HELLO.CPP for information on compiling.

6.5 HelloCS

HelloCS is the same as the Hello example program, except that it is written for Visual C#.

Load the project file HelloCS.csproj from the C# Development Environment.

6.6 AUTH_C

AUTH_C is a console mode C/C++ program that operates as a client implementing authenticated connections.

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

6.7 AUTH_S

AUTH_S is a console mode C/C++ program that operates as a server implementing authenticated connections.

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

6.8 CLIENT

CLIENT is a console mode example C/C++ program that operates as a client that connects to one of the example server programs.

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

6.9 ClientCS

ClientCS is an example GUI mode client program for Visual C# that connects to one of the example server programs. It is the C# version of vc_w_client.cpp.

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

6.10 Control

Control is an example console mode client that connects to a relay device that is controlled by sending commands to its TCP IP address..

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

6.11 Download

Download is an example console mode client that connects to the MarshallSoft web site (HTTP server) and downloads a zip-file from the web server.

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

6.12 FileSrv (“File Server”)

The FileSrv example server console mode program can accept multiple connections and allows clients to download files. See the comments in FileSrv.c or see PROTOCOL.TXT for a description of the protocol used.

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

6.13 FileCli (“File Client”)

The FileCli example client console mode program connects to the FileSrv example server program in order to download files. See the comments in FileCli.c or see PROTOCOL.TXT for a description of the protocol used.

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

6.14 FileGet

FileGet is a console mode C/C++ program that operates as a server, and receives files from the FileGet client. Files are decrypted when received.

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

6.15 FilePut

FilePut is a console mode C/C++ program that operates as a client, and transmits files to the FilePut server. Files are encrypted when transmitted.

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

6.16 GetPrice

GetPrice is an example console mode client that demonstrates how to download price quotes from Yahoo.

See the COMPILE comment in GetPrice.c for information on compiling

6.17 LIST

LIST is a console mode C/C++ program that displays all NIC (Network Interface Card) addresses on a given machine.

See the COMPILE comment in LIST.C for information on compiling.

6.18 POP3Stat

POP3Stat is an example console mode client that logs onto a POP3 account and returns the number of emails waiting.

See the COMPILE comment in POP3Stat.c for information on compiling.

6.19 Proxy

Proxy is a POP3 proxy console mode program. It is designed to accept connections from a POP3 email client and connect to a POP server, acting as a “go between” passing data in both directions.

The purpose of the Proxy example program is to demonstrate how to write proxy programs using **CSC**.

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

6.20 SERVER

SERVER is a console mode example C/C++ program that operates as a server that accepts connections from the example client program (CLIENT).

SERVER accepts a maximum of one connection at any one time. See SERVER2 for an example that can accept more than one connection at a time.

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

6.21 SERVER2

SERVER2 is a console mode example C/C++ program that operates as a server that accepts connections from the example client program (CLIENT).

The SERVER2 program uses a state driver to keep track of the state of multiple connections concurrently.

Edit the lines following "**** PROGRAMMER" in SERVER2.C before compiling. See the COMPILE comment in SERVER2.C for information on compiling.

6.22 uEcho_s

uEcho_s is an example UDP console mode echo server program. Whatever is sent to it is echoed back to the client. Port 7 is used by the echo server.

6.23 uEcho_c

uEcho_c is an example UDP console mode echo client program. Connect to any UDP echo server on port 7.

6.24 uNetTime

uNetTime is an example UDP console mode program that connects to a Network Time Server and receives the current network time (seconds since 1 January 1900 GMT).

Port 37 is used by Network Time Protocol servers. The default server is **time-A.timefreq.bldrdoc.gov**

6.25 VC_CLIENT

VC_CLIENT is an example console mode client program for Visual C++ .NET that connects to one of the example server programs.

See the COMPILE comment in VC_CLIENT.CPP for information on compiling.

6.26 VC_SERVER

VC_SERVER is a example console mode server program for Visual C++ .NET, which handles concurrent connections. Edit the lines following "*** PROGRAMMER" in VC_SERVER.CPP before compiling.

See the COMPILE comment in VC_SERVER.CPP for information on compiling.

6.27 VC_W_CLIENT

VC_W_CLIENT is an example GUI mode client program for Visual C++ .NET that connects to one of the example server programs. Edit the lines following "*** PROGRAMMER" in VC_W_CLIENT.CPP before compiling.

See the COMPILE comment in VC_W_CLIENT.CPP for information on compiling.

6.28 VC_W_SERVER

VC_W_SERVER is an example GUI mode server program for Visual C++ .NET that handles concurrent connections.

See the COMPILE comment in VC_W_SERVER.CPP for information on compiling.

6.29 W_SERVER

W_SERVER is a GUI mode C/C++ server program that operates as a server that can accept connections from the example client program (CLIENT) that handles multiple concurrent connections. Edit the lines following "*** PROGRAMMER" in W_SERVER.C before compiling.

See the COMPILE comment in W_SERVER.C for information on compiling.

6.30 W_CLIENT

W_CLIENT is a GUI mode C/C++ client program that operates as a client that connects to one of the example server programs. Edit the lines following "*** PROGRAMMER" in W_CLIENT.C before compiling.

See the COMPILE comment in W_CLIENT.C for information on compiling.

6.31 BCB_PRJ, BCB_PGM

BCB is a Borland C++ Builder Example program. Note that it is necessary to link with CSC32BCB.LIB rather than BCB32.LIB.

Load project file BCB_PRJ.BPR if running BCB version 4 or above, or BCB_PRJ.MAK if running BCB version 1 through 3.

7 Revision History

Version 1.2: March 3, 2004.

- The initial public release of CSC4C.

Version 2.0: September 24, 2004.

- Added cscChallenge and cscResponse.
- vSocks released if connection fails.
- Added cscWriteSocket.
- cscChallenge returns 8, not challenge value.
- Consider empty filename string same as NULL in cscGetFile.

Version 3.0: August 31, 2005.

- Corrected problem not freeing socket when CSCisConnect 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.

Version 4.0: March 28, 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: September 24, 2008

- cscSetInteger(Socket, 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 10, 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 August 29, 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 MakeDotted4 to make dotted IP address from it's 4 components.
- Added CONTROL.C example program.

[END]