Client / Server

Communications

Programmer's Manual

(CSC_4C)

Version 7.1

January 11, 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 C/C++ (CSC4C)** is a toolkit that allows software developers to quickly develop server and client TCP/IP and UDP applications using C/C++.

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. **CSC** can be used with our AES Advanced Encryption Library (**AES4C**) if strong encryption is desired.

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.

The **Client / Server Communications Library** runs under all 32-bit and 64-bit versions of Windows through Windows 10. The **Client / Server Communications Library SDK** DLLs (CSC32.DLL and CSC64.DLL) can also be used from any language (Visual Basic, Delphi, Visual FoxPro, COBOL, Xbase++, dBase, Microsoft Office, etc.) capable of calling the Windows API.

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)

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.
- Use with the MarshallSoft Advanced Encryption Standard (AES) 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.
- 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. (30 day). No unlock code is required.
- Can be used from GUI mode or console mode programs.
- Is fully thread safe.
- Supports 32-bit and 64-bit Windows though Windows 10.
- 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 2013).
- Works with Microsoft Visual Studio .NET and Visual C#.
- Works with Borland C/C++ (v5.0, v5.5, and Borland C++ Builder [all versions]).
- Works with Microsoft Foundation Class, Watcom v11, and MinGW GCC.
- Works with LCC-Win32 and Digital Mars
- 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 selection of C/C++ (also Visual C++ .NET and C# .NET) example programs with full source code is included. Refer to Section 5 for more details on each of the example programs.

Legend:

CON = Console mode.
GUI = GUI mode.
VS = Visual Studio C/C++
C# = Visual Studio C# (C Sharp)

auth_c.c	Authenticating client program (CON).
auth_s.c	Authenticating server program (CON).
bcb_prj.cpp	Displays CSC version, build, & registrations (Borland C++ Builder)
client.c	Client program (CON).
control.c	Controls device by sending commands to its TCP IP address (CON).
cscver.c	Displays CSC version, build, & registrations (CON).
download.c	Downloads a file from HTTP (web) sever (CON)
FileCli.c	File client connects to FileSrv to download files. (CON).
FileGet.c	Received files – crypto example (CON).
FilePut.c	Transmits files – crypto example (CON).
FileSrv.c	File server handles concurrent connections (CON).
hello.cpp	Uses C++ class wrapper to display version, build, etc. (CON).
HelloCS.cs	Displays CSC version, build, and registrations screen (C#)
List.c	Lists all TCP/IP addresses (CON).
POP3Stat.c	Returns the number of emails waiting on POP3 server (CON).
Proxy.c	POP3 proxy program (CON)
Server.c	Server program (CON)
server2.c	Server program. Handles concurrent connections (CON).
uEcho_c.c	Example UDP echo client (CON).
uEcho_s.c	Example UDP echo server (CON).
uNetTime.c	Gets network time from UDP Network Time Server (CON).
vc_client.cpp	Client program (CON: VS).
vc_server.cpp	Server program. Handles concurrent connections (CON: VS).
vc_vers.cpp	Displays CSC version, build, and registrations screen (CON: VS).
vc_w_client.cpp	Client program (GUI: VS
vc_w_server.cpp	Server program. Handles concurrent connections (GUI: VS).
vers.c	Displays CSC version, build, & registrations (GUI).
w_client.c	Client program (GUI).
w_server.c	Server program. Handles concurrent connections (GUI).
SendUDP.c	Transmits a UDP data packet to RecvUDP (CON).
RecvUDP.c	Receives multicast UDP packets from SendUDP (CON).
vc_SendUDP.cpp	Transmits a UDP data packet to vc_RecvUDP (CON: VS)
vc_RecvUDP.cpp	Receives multicast UDP packets from vc_SendUDP (CON: VS)

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 (CSC_4C.PDF) 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 <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include "csc.h"

// connect to TCP server and read it's greeting message

int ConnectToServer(char *Server, int Port, char *Buffer, int BufLen)
{int Code;
 int DataSock;
 // the client connects to the server
 DataSock = cscClient(Server, Port);
 // error? (negative return codes are errors)
 if(DataSock<0) return DataSock;
 // wait (3 sec) for greeting message from server
 if(cscAwaitData(DataSock, 3000))
    {// read server's greeting message into Buffer[]
     Code = cscGetData(DataSock, Buffer, BufLen);
     if(Code<0) return Code;
     // terminate server's text message with a NULL
     Buffer[Code] = '\0';
     // return (virtual) data socket
     return DataSock;
    }
 // timed out
 return CSC_CONNECT_TIMEOUT;
}
```

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 CSC4C71.ZIP (evaluation version) or CSCxxxxx.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.

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

1.6 Pricing

A developer license for the Client/Server Communications Library can be purchased for $119 (or $199 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).

Also see INVOICE.TXT or https://www.marshallsoft.com/vmorder.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 (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

- $33 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).
- $77 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 XP through Windows 10.

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 DLLs
```

2.1 Library Files

The **Client/Server Communications Library** (CSC) is distributed as a set of DLL's (CSC32.DLL and CSC64.DLL) and a set of object files (CSC32.OBJ and CSC64.OBJ) for static linking.

In order to use the DLLs, link with the lib files CSC32.LIB or CSC64.DLL.

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 Console Mode & GUI Mode

CSC functions can be called from both console mode and from GUI (Graphical User Interface) mode Windows programs. **CSC** functions are called in exactly the same way in both cases.

2.4 Static Linking

The user can statically link CSC32.OBJ (or CSC64.OBJ) with their application, rather than linking with CSC32.LIB (or CSC64.LIB).

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

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, if source code was purchased 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 from

```
#ifndef STATIC_LIBRARY
```

through the following

```
#endif
```

2.5 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.

2.6 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

See the makefiles and project files for the example programs.

2.7 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)

2.8 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 it will always be found by Windows at run time.

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.

2.9 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 "	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.

2.10 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.

2.10.1 Visual Studio C/C++: Versions 2005 through 2015

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).

2.10.2 Visual Studio Visual Basic: Versions 2005 through 2015

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".

2.11 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 , 2010, 2012, and 2013 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

vc_FileGet(VS2012)x64.vcxproj     vc_FilePut(VS2012)x64.vcxproj
vc_server(VS2012)x64.vcxproj      vc_client(VS2012)x64.vcxproj
vc_w_client(VS2012)x64.vcxproj    vc_w_server(VS2012)x64.vcxproj
vc_vers(VS2012)x64.vcxproj

vc_FileGet(VS2013)x64.vcxproj     vc_FilePut(VS2013)x64.vcxproj
vc_server(VS2013)x64.vcxproj      vc_client(VS2013)x64.vcxproj
vc_w_client(VS2013)x64.vcxproj    vc_w_server(VS2013)x64.vcxproj
vc_vers(VS2013)x64.vcxproj
```

3 Supported Compilers

The **Client/Server Communications Library for C/C++ (CSC4C)** has been tested and works with all versions of the following compilers:

- Microsoft Visual C/C++ (32-bit)
- Visual Studio (through VS 2015)
- Borland C/C++
- Borland C++ Builder
- Watcom C/C++
- MinGW GCC
- LCC-Win32/Win64
- Digital Mars

3.1 Microsoft Visual C/C++ (VC 4.0 through 6.0)

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 (version 4.0 through 6.0) compile Win32 programs ONLY. CSC does not support Win16.

Visual C/C++ project files use file extension ".mak". Visual C/C++ 6.0 project files use file extension ".dsp"

Visual C/C++ makefiles use file extension "._M_" and are contained in the file makefiles(microsoft).zip.

3.2 Microsoft Visual Studio C/C++

CSC4C works with all versions of Visual Studio

3.2.1 Microsoft Visual Studio C/C++ 2003 and 2005

Visual Studio 2003 through 2005 project files use file extension ".vcproj"

3.2.2 Microsoft Visual Studio C/C++ 2008

Visual Studio 2008 specific project files end with ".(VS2008).vcproj" for 32-bit applications and ".(VS2008)x64.vcproj" for 64-bit applications. Visual Studio 2008 can compile both 32-bit and 64-bit programs.

3.2.3 Microsoft Visual Studio C/C++ 2010

Visual Studio 2010 specific project files end with ".(VS2010).vcxproj" for 32-bit applications and ".(VS2010)x64.vcxproj" for 64-bit applications.

Note that project files for Visual Studio 2010 end with ".vcxproj" rather than ".vcproj" as in earlier versions of Visual Studio.

3.2.4 Microsoft Visual Studio C/C++ 2012

Visual Studio 2012 specific project files end with ".(VS2012).vcxproj" for 32-bit applications and ".(VS2012)x64.vcxproj" for 64-bit applications.

In order to convert an older project file to VS 2012:

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

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

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

3.2.5 Microsoft Visual Studio C/C++ 2013 & 2015

Visual Studio 2013 specific project files end with ".(VS2013).vcxproj" for 32-bit applications and ".(VS2013)x64.vcxproj" for 64-bit applications.

Projects can be converted from VS 2010 as described in the previous section.

3.3 Microsoft C/C++ Express Edition

The “Express Edition” of Microsoft Visual Studio is available as a free download at
http://www.microsoft.com/express/download/

Open the VC project file as in previous versions of Visual Studio.

3.4 Microsoft Studio 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 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, System.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.

3.5 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. All Borland 5.0 makefiles are contained in file `makefiles(borland50).zip`.

Borland C/C++ Version 5.5 (which can be downloaded from http://edn.embarcadero.com/article/20633) 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!

All Borland 5.5 makefiles are contained in file `makefiles(borland55).zip`.

Borland programs always link with CSC32BCB.LIB.

3.6 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.

3.7 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.

All Watcom 11 makefiles use file extension "._W_" and are contained in file `makefiles(watcom11).zip`

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]".

3.8 LCC-Win32/Win64 C/C++

LCC-Win32/Win64 C programs can be compiled from either the command line or from within the development environment.

LCC-Win32/Win64 is a freeware C compiler developed and distributed by Jacob Navia at

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

All LCC-Win32/Win64 makefiles use file extension "._L_" are contained in file makefiles(lcc).zip

3.9 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

All MinGW GCC makefiles use file extension "._G_" are contained in file makefiles(gcc).zip

3.10 Digital Mars C/C++

All Digital Mars makefiles use file extension "._D_" are contained in file makefiles(mars).zip

4 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 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 Example Programs

Example programs include both console mode and GUI examples, and are located in the APPS sub-directory.

Makefiles and project files are classified as follows:

- `*._M_` Microsoft C/C++ makefiles.
- `*._B_` Borland C/C++ 5.0 makefiles.
- `*._I_` Borland C/C++ 5.5 makefiles.
- `*._W_` Watcom C/C++ makefiles.
- `*._L_` LCC Win32/Win64 makefiles.
- `*._G_` MinGW GCC makefiles.
- `*._D_` Digital Mars makefiles.
- `*.dsp` Microsoft v6.0 project file.
- `*.vcproj` Microsoft Visual C++ project file.
- `*.vcxproj` Microsoft Visual C++ project file (VS2010 thru VS2013).
- `*.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.PDF) for basic information on Server / Client protocols.

Makefiles and projects can be found in the \CSC4C\APPS sub-directory as follows:

- makefiles(microsoft).zip all Microsoft makefiles.
- makefiles(borland50).zip all Borland 5.0 makefiles.
- makefiles(borland55).zip all Borland 5.5 makefiles.
- makefiles(gcc).zip all MinGW GCC makefiles.
- makefiles(lcc).zip all LCC Win32/Win64 makefiles,
- makefiles(watcom11).zip all Watcom 11 makefiles.
- makefiles(mars)/.zip all Digital Mars makefiles.

- Projects(VS2008).zip all Visual Studio 2008 project files.
- Projects(VS2010).zip all Visual Studio 2010 project files.
- Projects(VS2012).zip all Visual Studio 2012 project files.
- Projects(VS2013).zip all Visual Studio 2013 project files.

5.1 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.

5.2 VERS

VERS is a Windows (GUI mode) example C/C++ program that displays the CSC32.DLL (CSC64.DLL) Version, Build #, and registration string.

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

5.3 VC_VERS

VC_VERS is a Visual C++ .Net Windows (GUI mode) example program that displays the CSC32.DLL (CSC64.DLL) Version, Build #, and registration string.

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

5.4 HELLO

HELLO is a console mode example C++ program that displays the CSC32.DLL (CSC64.DLL) Version, Build #, and registration string using the **CSC** class (fcsc.cpp).

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

5.5 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.

5.6 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.

5.7 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.

5.8 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.

5.9 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.

5.10 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.

5.11 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.

5.12 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.

5.13 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.

5.14 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.

5.15 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.

5.16 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.

5.17 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.

5.18 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.

5.19 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.

5.20 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.

5.21 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.

5.22 uEcho_c

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

5.23 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**

5.24 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.

5.25 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.

5.26 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.

5.27 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.

5.28 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.

5.29 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.

5.30 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.

5.31 SendUDP

SendUDP is a console mode program that transmits a UDP data packet. It is used to test the RecvUDP.c program.

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

5.32 RecvUDP

RecvUDP is a console mode program that receives multicast UDP data packets. Test using the SendUDP.c program.

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

5.33 vc_SendUDP

SendUDP is a Visual C++ console mode program that transmits a UDP data packet. It is used to test the RecvUDP.c program.

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

5.34 vc_RecvUDP

RecvUDP is a Visual C++ console mode program that receives multicast UDP data packets. Test using the SendUDP.c program.

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

6 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(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 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 its 4 components.
- Added CONTROL.C example program.

Version 6.2: February 15, 2011

- Added function cscMakeDotted4()
- Fixed Julian date function
- Added additional diagnostics (to detect congestion) in csc-vs.c
- Fixed cscReadSize(), previous version worked only for vSock 0
- Fixed CSC_SET_TIMEOUT_VALUE problem
- 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: June 13, 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 returning bogus IP addresses after the first.

Version 7.0: March 12, 2015

- Added cscMulticast() that receives multicast UDP packets.
- Added cscClientExt() that binds to a local IP address (for multi-homed computers)
- Added additional makefiles (MS, Borland, Watcom, MinGW GCC, LCC, Digital Mars) for example programs.
- Added Microsoft VS2012 and VS2013 project files.
- Fixed bogus CSC_BAD_OFFSET error sometimes returned by crcCryptoPutFile().

Version 7.1: January 11, 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.