

Client / Server Communications

Library for Delphi

Programmer's Manual

(CSC4D)

Version 6.2

March 5, 2012

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

Copyright (C) 2012
All rights reserved

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

Voice: 1.256.881.4630
Web: <http://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 Libraries	Page 8
2.2	Keycode	Page 8
2.3	Win32 STDCALL and DECLSPEC	Page 8
2.4	Adding CSC4D to Your Project	Page 9
2.5	CSC4D Wrapper	Page 9
2.6	Example Protocol	Page 10
2.7	Error Display	Page 10
3	Compiler Issues	Page 11
3.1	Delphi Versions	Page 11
3.2	Compiling Programs	Page 13
3.3	Compiling CSC Source	Page 13
4	Delphi Example Programs	Page 14
5	Revision History	Page 18

1 Introduction

The **Client / Server Communications Library for Delphi (CSC4D)** is a toolkit that allows software developers to quickly develop server client TCP/IP and UDP applications in Delphi and Delphi for .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]).

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

The **Client / Server Communications Library for Delphi (CSC4D)** component library supports and has been tested with all 32-bit and 64-bit versions of Delphi including:

- Borland Delphi (2.0, 3.0, 4.0, 5.0, 6.0 and 7.0)
- Borland Delphi 8 for .NET
- Borland Delphi 2005 & 2006
- Borland Turbo Delphi
- Codegear Delphi 2007
- Embarcadero Delphi 2009 & 2010, Embarcadero Delphi XE and Delphi XE2

CSC4D includes multiple Delphi example programs demonstrating client/server protocols, including examples that connect to HTTP (web) and POP3 servers as well as encrypt files.

CSC4D runs under all versions of Windows (95/98/ME/2000/2003/NT/XP/Vista/Win7). The **Client / Server Communications Library SDK DLLs (CSC32.DLL or CSC64.DLL)** can also be used from any language (Visual C++, .NET, Visual Basic, VB.NET, ACCESS, EXCEL, VBA, Visual FoxPro, COBOL, Xbase++, dBase, etc.) capable of calling the Windows API.

The **Client/Server Communications Programmer's Manual** provides information needed to compile programs using **CSC** in a Delphi programming environment.

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

- **CSC** is a standard Windows DLL (NOT an OCX or ActiveX control).
- **CSC** does NOT depend on ActiveX or Microsoft Foundation Class (MFC) or similar "support" libraries.
- **CSC** is fully thread-safe.
- **CSC** functions can be called from applications not capable of using controls.

MarshallSoft also has versions of the **Client/Server Communications Library** for C/C++ (**CSC4C**), Visual Basic (**CSC4VB**), dBase (**CSC4DB**), Xbase++ (**CSC4XB**) and Visual FoxPro (**CSC4FP**). Each version of **CSC** uses the same DLLs (**CSC32.DLL** or **CSC64.DLL**). However, the examples provided for each version are written for the specified programming language.

All versions of the **Client/Server Communications Library (CSC)** can be downloaded from our web site at

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

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

1.1 Features

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

- Supports both UDP and TCP protocols.
- Supports both Win64 and Win32.
- 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.
- Servers can handle multiple connections concurrently.
- Supports secure and private messaging.
- Create chat server and clients.
- Create client / server file transfer
- Create client programs to talk to TCP servers (POP3, IMAP, HTTP, SMTP, DNS)
- Create SMTP proxy programs extracting a copy of all recipient addresses
- Create POP3 proxy programs that filter incoming email for Spam
- Create HTTP proxy used to filter content
- 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
- Allows multiple servers and clients to run simultaneously.
- Royalty free distribution with your compiled application.
- Evaluation versions are fully functional. No unlock code is required.
- Is fully thread safe.
- Supports Windows 95/98/Me/NT/2000/2003/XP/Vista/Windows 7.
- Works with all versions of Delphi including Delphi 2005 through 2007, Delphi for .NET, Embarcadero Delphi 2009 - 2010, Embarcadero Delphi XE, and Embarcadero Delphi XE2.
- Does not depend on support libraries. Makes calls to core Windows API functions only.
- Can be used with any program (in any language) capable of calling Windows API functions.
- Can be purchased with (or without) ANSI C source code.
- Updates are free for one year (Source code updates are separate).
- License covers all programming languages (C/C++, VB, Delphi, etc).
- Free unlimited one-year technical support for registered users.
- Documentation online as well as in printable format.

A selection of Delphi (Borland /CodeGear/Embarcadero) example programs with full source code is included. Refer to Section 6 for more details on each of the example projects.

ver_prj.dpr	Displays CSC version and build.
helo_prj.dpr	Displays CSC version and build using wrapper.
client_prj.dpr	Example client program.
server_prj.dpr	Example server program.
download_prj.dpr	Example download (from web) program.
auth_client_prj.dpr	Authenticating client.
auth_server_prj.dpr	Authenticating server.
File_Server_prj.dpr	Example file server.
File_Client_prj.dpr	Example file client.
POP3Stat.dpr	Returns the number of emails waiting on POP3 server.
GetPrice.dpr	Downloads stock price from finance.yahoo.com
uNetTime.dpr	UDP client gets Network Time.
Client.bdsproj	Example Delphi NET client program.
Server.bdsproj	Example Delphi NET server program.
Control_prj.dpr	Controls device; sends commands to its IP address.

1.2 Documentation Set

This is the first manual (CSC_4D) in the set.

- [CSC4D Programmer's Manual](#) (CSC_4D.PDF)
- [CSC User's Manual](#) (CSC_USR.PDF)
- [CSC Reference Manual](#) (CSC_REF.PDF)

The CSC_4D Programmer's Manual is the language specific (Delphi) manual. All language dependent programming issues are discussed in this manual. Information needed to compile programs in a Delphi environment is provided in this manual.

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

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

The documentation is also provided on our web site at

<http://www.marshallsoft.com/csc4d.htm>

1.3 Example Program

The following code segment attempts to connect to the server.

```
{connect to server}
DataSock := cscClient(HostName, HostPort);
if DataSock < 0 then
begin
    DisplayLine(client.mHistory, 'cscClient fails');
    DisplayError(client.mHistory, DataSock);
    exit;
end;
{connected: now wait 7 seconds for greeting message from server}
Call DisplayText(client.mHistory, "Awaiting greeting message...")
Code := cscAwaitData(DataSock, 7000);
if Code = 0 then
begin
    DisplayLine(client.mHistory, 'Greeting message not seen');
    exit;
end;
```

Also see the example programs in the \CSC4D\APPS sub-directory.

1.4 Installation

- (1) Before installation of CSC4D, a Delphi compiler should already be installed.
- (2) Unzip CSC4D62.ZIP (evaluation version) or CSCxxxxx.ZIP (registered version: xxxxx is the Customer ID) using any Windows unzip program.
- (3) Run the installation program SETUP.EXE which will install all CSC4D 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 CSC4D is very easy. First, delete the CSC4D project directory created when installing CSC4D. Second, delete CSC32.DLL/CSC64.DLL from the Windows directory, typically C:\WINDOWS for Windows 95/98/Me/2003/XP/VISTA/Win7 or C:\WINNT for Windows NT/2000.

1.6 Pricing

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

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

1.7 Updates

When a developer license is purchased for CSC, the developer will be sent a registered DLL plus a license file (CSCxxxxx.LIC). The license file can be used to update the registered DLL for a period of one year from purchase. Updates can be downloaded from

<http://www.marshallsoft.com/oem.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 1 and 3 years of the original purchase (or previous update).
- \$75 if the update is ordered after three years of the original purchase (or previous update).

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

2 CSC Library Overview

The **Client/Server Communications Library (CSC)** has been tested on multiple computers running Windows 95/98, Windows Me, Windows NT4, Windows 2000, Windows 2003, Windows XP, Windows Vista and Windows 7.

The CSC4D library will work with all versions of Delphi (Borland/CodeGear/Embarcadero) including Delphi for .NET, Delphi 2010, Embarcadero Delphi XE and Embarcadero Delphi XE2. The CSC32.DLL/CSC64.DLL functions may be called by any Windows application program capable of calling the Windows API provided that the proper declaration file is used.

The SETUP installation program will copy the Lib's and DLLs to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the CSC4D files are copied to the directory specified (default \CSC4D). Three sub-directories are created, as follows:

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

2.1 Dynamic Link Libraries

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

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

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

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

2.2 Keycode

CSC32.DLL has 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.PAS. The keycode for the evaluation version is 0. You will receive a new key code when registering. The KEYCODE is passed to **cscAttach**.

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

2.3 Win32 STDCALL and DECLSPEC

CSC is written in ANSI C and is compiled using the `_stdcall` and `_declspec` keywords. This means that CSC uses the same calling conventions and file naming conventions as the Win32/Win64 API. In particular, function names are NOT decorated. There are no leading underscores or trailing "@size" strings added to function names.

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

2.4 Adding CSC4D to Your Project

Copy CSC32.PAS (or CSC64.PAS) to the same directory (folder) as the application program to which you want to add CSC code. The files will be in the \APPS directory (folder) created when SETUP was run, usually C:\CSC4D\APPS.

For Win32 Delphi, add CSC32 and keycode to the "uses" clause in the application source program (*.PAS). For example,

```
uses
    csc32, keycode, ...
```

Alternatively, the above "keycode" can be left out and instead the numerical keycode value (found in keycode.pas) can be pasted directly into the call to cscAttach. Also add csc32 to the project file (*.DPR). For example,

```
uses
    csc32 in 'csc32.pas', ...

    {pass the key code [0 for evaluation version]}

    Code := cscAttach(1, 0, 123456789);
```

2.5 CSC4D Wrapper

The CSC32W.PAS module is a "wrapper" for CSC32.PAS that allows you to pass Delphi strings directly rather than having to first convert them to PAnsiChar variables. Strings in CSC, like Windows itself, are in reality buffers terminated by a null (0) character. Implementing CSC in this way allows CSC functions to be called by any language capable of calling the Windows API. Note that wrapper functions (defined in CSC32W.PAS) are named beginning with the letter 'f'. For instance, "**fAttach**" is the wrapper function that calls "**cscAttach**".

Nevertheless, it is certainly more convenient to pass Delphi strings rather than zero terminated buffers. Compare the two following code segments:

```
{call using CSC -- pass pointers to null terminated buffers}
var
    Code      : Integer;
    ServerPtr : PChar;
    ServerPort : Integer;
begin
    GetMem(ServerPtr, 128);
    StrPCopy(ServerPtr, 'mail.marshallsoft.com');
    ServerPort := 5001;
    Code := cscClient(ServerPtr, ServerPort);
    FreeMem(ServerPtr, 128);
    . . .
    {call using CSC32W -- pass Delphi strings}
    . . .
    Code := fClient('mail.marshallsoft.com', 5001);
    . . .
```

2.6 Example 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	6.1
BYE	OK (then disconnects)	BYE	OK
ECHO	Sends string after "ECHO "	ECHO Hello	Hello

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

Also refer to PROTOCOL.TXT in the \CSC4D\APPS directory.

2.7 Error Display

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

```
procedure DisplayError(Memo: TMemo; ErrCode : Integer);
var
  Code : Integer;
  Text : String;
  Ptr : PChar;
begin
  if ErrCode < 0 then
    begin
      GetMem(Ptr, 128);
      Code := cscErrorText(ErrCode, Ptr, 127);
      if Code > 0 Then DisplayLine(Memo, StrPas(Ptr));
      Text := Format('CSC error %d.', [ErrCode]);
      FreeMem(Ptr, 128);
      DisplayLine(Memo, Text);
    end
  end;
end;
```

3 Compiler Issues

The **Client/Server Communications Library for Delphi** component library supports all versions of Borland/CodeGear /Embarcadero) Delphi for Win32/Win64 as well as Delphi for .NET as follows:

- Borland Delphi 2, 3, 4, 5, 6, 7 and 8.
- Borland Delphi 2005 (Delphi 9)
- Borland Delphi 2006
- Borland Turbo Delphi
- CodeGear Delphi 2007
- CodeGear Delphi 2009
- CodeGear Delphi 2010
- Embarcadero Delphi XE
- Embarcadero Delphi XE2

3.1 Delphi Versions

Applications written with Delphi link with the same identical DLL's as for applications written in all other supported languages, such as C/C++ and Visual Basic.

Beginning with Delphi 2003, Delphi has two "personalities": (1) Win32 Delphi and (2) Delphi for .NET. Win32 Delphi programs are a continuation of the Delphi language as seen in earlier versions of Delphi. Delphi .NET is a version of Delphi designed to use the Microsoft .NET Framework.

3.1.1 Delphi 1

Delphi version 1 generates Win16 code. CSC32.DLL is a 32-bit DLL and cannot be called from 16-bit application programs such as those compiled by Delphi 1.

3.1.2 Delphi 2

Delphi version 2 and above generates WIN32 code. Therefore, applications written using Delphi 2 will link with CSC32.DLL. Strings can be much larger than 255 bytes.

Delphi 2 seems to have a problem with some of the string functions. Although the default is "large strings", some of the string functions (such as StrPas) copy only 255 bytes.

3.1.3 Delphi 3, 4, 5, and 6.

There are no known Delphi problems impacting our example programs in Delphi version 3 and above. Applications written using Delphi 3 and above will link with CSC32.DLL

3.1.4 Delphi 7.

Beginning in Delphi 7, the filename of a unit must match the unit name.

3.1.5 Delphi 2005 through 2010

Delphi 2005 through Delphi 2009 are Borland's (Codegear) latest Delphi product with support for both Win32 and Microsoft .NET Framework. Application programs written using Delphi 2005, 2006, 2007, 2009, 2010, and XE will use CSC32.DLL.

When loading Win32 Delphi projects with Delphi 2005/2006/2007/2009/2010/XE, a window entitled "Project Upgrade" may be displayed. In this case, the project must be upgraded before it can be opened:

```
Please select which project type you wish to target:  
( ) Delphi for .NET  
( ) Delphi for Win32
```

Choose "Delphi for Win32" for all projects except "client.bdsproj" and "server.bdsproj", which are Delphi for .NET projects.

3.1.6 Delphi XE2

Delphi XE2 can create both 32-bit and 64-bit executables. Note that although a 32-bit executable can run on both 32-bit and 64-bit Windows machines, a 64-bit executable can run only on a 64-bit Windows machine.

Step 1: Edit Files

1. Open file *_PGM.PAS using any text editor.
2. Replace CSC32 with CSC64 in the "uses" clause.
3. Open file *_PRJ.DPR using any text editor.
4. Replace CSC32 with CSC64.

Step 2: Delete Win32 files

1. Delete all *.DCU files that may have been created under Win32 Delphi.
2. Delete all *.DPROJ files that may have been created under Win32 Delphi.

Step 3: Change Win32 to Win64 in Delphi XE2 Project Manager

Start Delphi XE2, load the project *_PRJ.DPR, then open the Delphi Project Manager.

1. Right click on "Target Platforms (Win32)" and click on "Add Platform...".
2. When the "Select Platform" window is displayed, click "64-bit Windows".
3. Click [+] box to left of "Target Platforms (Win32)"
4. Right click on "32-bit Windows", then click "Remove Platform"

You must also convert display.pas by replacing all reference to csc32 with csc64.

The file x64.zip contains the example programs after modification to x64.

3.2 Compiling Programs

The example programs are compiled from the Delphi development environment using the provided Delphi project files (*.DPR).

The example programs will compile and run with any version of Delphi. They have each been tested using Delphi 1 through Delphi 5, Delphi 2005 through Delphi 2007, Embarcadero Delphi 2009 & 2010 and Embarcadero Delphi XE & XE2. The file x64.zip contains the example programs after modification to x64.

Respond with "OK" to the message "Cannot find resource file..." and it will be properly rebuilt.

See Section 4 "Example Programs" for more details on each of the example programs.

3.3 Compiling CSC Source

The source code for the CSC DLL's is written in standard ANSI C (CSC.C), and has been compiled using Microsoft Visual C++. Source code for the CSC library can be purchased at the same time as a CSC developer license is purchased.

CSC may also be compiled using Borland C/C++ or Watcom C/C++ compilers. If CSC.C is compiled using Borland or Watcom compilers, the resulting DLL can only be used by applications compiled with the same compiler, unless the STDCALL and DECLSPEC keywords are specified.

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

4 Example Programs

The DISPLAY.PAS unit is used to display text in Delphi memos. DISPLAY.PAS contains 4 procedures:

```
DisplayChar   : Displays character.  
DisplayString : Displays string.  
DisplayLine   : Displays line.  
DisplayError  : Displays error message.
```

4.1 Delphi Example Programs

The example programs are designed to demonstrate the various capabilities of CSC4D. The best way to become familiar with CSC4D is to study and run the example programs.

The example programs can be compiled with any version of 32-bit (or 64-bit) Delphi although there are separate examples for Delphi NET. All example programs are located in the \CSC4D\APPS sub-directory.

The CSC units used by the example programs are:

```
CSC32.PAS      : Win32 CSC unit.  
CSC32W.PAS    : Win32 CSC unit used by HELO project.  
CSC32UC.PAS   : NET unit used by Client NET project.  
CSC64.PAS     : Win64 CSC unit.  
CSC64W.PAS    : Win64 CSC unit used by HELO project.
```

4.1.1 VERSION

The VERSION example program displays the CSC version number, build, and registration string. This is the first program to compile and build since it verifies that CSC32.DLL is installed properly.

The VERSION project files are:

```
VER_PRJ.DPR    : Delphi project file.  
VER_PGM.DFM    : Delphi form file.  
VER_PGM.PAS    : Program source code.
```

4.1.2 HELO

HELO is an example program that displays the CSC version number, build, and registration string using the CSC32W.PAS wrapper unit.

The HELO project files are:

```
HELO_PRJ.DPR   : Delphi project file.  
HELO_PGM.DFM   : Delphi form file.  
HELO_PGM.PAS   : Program source code.
```

4.1.3 CLIENT

CLIENT is an example client program that operates as a client that connects to the example server program (SERVER).

The CLIENT project files are:

```
CLIENT_PRJ.DPR : Delphi project file.  
CLIENT_PGM.DFM : Delphi form file.  
CLIENT_PGM.PAS : Program source code.
```

4.1.4 SERVER

SERVER is an example server program that accepts connections from the example client program (CLIENT).

SERVER accepts a maximum of 3 connections (clients) at any one time.

The SERVER project files are:

```
SERVER_PRJ.DPR : Delphi project file.  
SERVER_PGM.DFM : Delphi form file.  
SERVER_PGM.PAS : Program source code.
```

4.1.5 File_Server

The File_Server example server program can accept multiple connections and allows clients to download files. Files are encrypted when transmitted. See the comments in File_Server_Pgm.pas or see PROTOCOL.TXT for a description of the protocol used.

The File_Server project files are:

```
File_Server_PRJ.DPR : Delphi project file.  
File_Server_PGM.DFM : Delphi form file.  
File_Server_PGM.PAS : Program source code.
```

4.1.6 File_Client

The File_Client example client program connects to the File_Server example server program in order to download files. Files are decrypted when received. See the comments in File_Client_Pgm.pas or see PROTOCOL.TXT for a description of the protocol used.

The File_Client project files are:

```
File_Client_PRJ.DPR : Delphi project file.  
File_Client_PGM.DFM : Delphi form file.  
File_Client_PGM.PAS : Program source code.
```

4.1.7 Auth_Server

The Auth_Server example program demonstrates how server-side authentication (challenge/response) is done in CSC4D. Auth_Server accepts connections from Auth_Client.

The File_Server project files are:

```
Auth_Server_PRJ.DPR : Delphi project file.  
Auth_Server_PGM.DFM : Delphi form file.  
Auth_Server_PGM.PAS : Program source code.
```

4.1.8 Auth_Client

The Auth_Client example program demonstrates how client-side authentication (challenge/response) is done in CSC4D. Auth_Server accepts connections from Auth_Client.

The File_Client project files are:

```
Auth_Client_PRJ.DPR : Delphi project file.  
Auth_Client_PGM.DFM : Delphi form file.  
Auth_Client_PGM.PAS : Program source code.
```

4.1.9 (NET) Client

The (NET) CLIENT example program is the Delphi 2005/2006/2007/2009/2010/XE for .NET equivalent of the Delphi Win32 CLIENT example program.

The Delphi 2005/2006/2007/2009/2010/XE NET project files are:

```
client.bdsproj      : Delphi NET project file  
client.dpr          : Delphi NET project file  
WinForm_Client.pas : Delphi NET form source  
WinForm_Client.resx : Delphi NET resource file  
WinForm_Client.TWinForm.resources : Delphi NET resource file
```

4.1.10 (NET) Server

The (NET) SERVER example program is the 2005/2006/2007/2009/2010/XE NET equivalent of the Delphi Win32 SERVER example program.

The 2005/2006/2007/2009/2010/XE NET project files are:

```
server.bdsproj      : Delphi NET project file  
server.dpr          : Delphi NET project file  
WinForm_Server.pas : Delphi NET form source  
WinForm_Server.resx : Delphi NET resource file  
WinForm_Server.TWinForm.resources : Delphi NET resource file
```

4.1.11 Download

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

The Download project files are:

```
Download_PRJ.DPR : Delphi project file.  
Download_PGM.DFM : Delphi form file.  
Download_PGM.PAS : Program source code.
```

4.1.12 GetPrice

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

The GetPrice project files are:

```
GetPrice_PRJ.DPR : Delphi project file.  
GetPrice_PGM.DFM : Delphi form file.  
GetPrice_PGM.PAS : Program source code.
```

4.1.13 POP3Stat

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

The POP3Stat project files are:

```
POP3Stat_PRJ.DPR : Delphi project file.  
POP3Stat_PGM.DFM : Delphi form file.  
POP3Stat_PGM.PAS : Program source code.
```

4.1.14 uNetTime

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

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

The uNetTime project files are:

```
uNetTime_PRJ.DPR : Delphi project file.  
uNetTime_PGM.DFM : Delphi form file.  
uNetTime.PGM.PAS : Program source code.
```

4.1.15 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..

The Control project files are:

```
Control_PRJ.DPR : Delphi project file.  
Control_PGM.DFM : Delphi form file.  
Control.PGM.PAS : Program source code.
```

5 Revision History

CSC is written in ANSI C. All language versions of CSC (C/C++, Visual Basic, and Delphi) use the same identical DLL.

Version 3.0: November 1, 2005

- Initial Delphi release of CSC.

Version 4.0: April 25, 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 "Days Left" (for evaluation version)

Version 5.0: October 9, 2008

- cscSetInteger(Socket, CSC_SET_LINGER, LingerTime) returns LingerTime
- Added cscLaunch()
- Increased NBR_LISTEN_SOCKS to 64.
- Added GetPrice 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: August 5, 2009

- Supports Win64 (although Delphi thru 2010 does not)
- 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(Channel, CSC_GET_SOCKET) returns actual socket.
- Added cscCreateUDP, cscGetUDP, cscCreateUDP.
- Added cscNetToHost32.

Version 6.1: September 8, 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 MakeDotted4 to make dotted IP address from its 4 components.
- Added CONTROL example program.

Version 6.2: March 5, 2012

- Added support for Delphi XE2
- Added function cscMakeDotted4()
- Corrected Julian date function
- Added additional diagnostics (to detect congestion) in csc-vs.c
- Fixed cscReadSize(), worked only for vSock 0.
- Fixed CSC_SET_TIMEOUT_VALUE not being passed to csc-vs
- Fixed cscChallenge uses leading zeros to pad to 8 chars
- Changed DEFAULT_PACKET_TIMEOUT to 35000
- Added error text for VS_* errors.
- Added functions cscPutFileExt & cscCryptoPutFileExt
- Added functions cscGetFileExt & cscCryptoGetFileExt
- Fixed problem receiving file with same name from two clients