

Client / Server Communications

Library for Visual dBASE

Programmer's Manual

(CSC4DB)

Version 7.1

January 26, 2018

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

Copyright (C) 2018
All rights reserved

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

Voice: 1.256.881.4630
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 5
1.3	Example Program	Page 5
1.4	Installation	Page 6
1.5	Uninstalling	Page 6
1.6	Pricing	Page 6
1.7	Updates	Page 6
2	CSC Library Overview	Page 7
2.1	Dynamic Link Libraries	Page 7
2.2	Keycode	Page 7
2.3	INCLUDE Files	Page 7
2.4	dBASE Forms	Page 8
2.5	Dynamic Strings	Page 8
2.6	Null Terminated Strings	Page 8
2.7	Adding CSC4DB to a dBASE Program	Page 9
2.8	Example Protocol	Page 9
2.9	Error Display	Page 9
3	Compiler Issues	Page 10
3.1	Compiling dBASE Programs	Page 10
3.2	Compiling dBASE Projects	Page 10
3.3	Creating Executables	Page 10
3.4	Compiling CSC Source	Page 10
4	dBASE Example Programs	Page 11
5	Revision History	Page 13

1 Introduction

The **Client / Server Communications Library for Visual dBASE (CSC4DB)** is a toolkit that allows software developers to quickly develop server and client TCP/IP and UDP applications in Visual dBASE and dBASE Plus.

The **Client / Server Communications Library (CSC)** is a component DLL library used to create **server** and **client** programs that can communicate with each other across any TCP/IP or UDP network such as the Internet or a private network (intranet or LAN [local area net]). The **CSC** component library uses the Windows API (Application Programmer's Interface) and Windows sockets API for all communication.

The **CSC** library can be used to communicate with other **CSC** programs or 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 Visual dBASE (CSC4DB)** component library supports all 32-bit (dBASE 7.0, 7.5 and dBASE Plus) versions of dBASE. **CSC4DB** includes several dBASE 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 (AES4DB)** if strong encryption is desired.

A Win32 DLL is provided (a 64-bit DLL is available). **CSC4DB** runs under all versions of 32-bit and 64-bit Windows XP through Windows 10 computers. The **Client / Server Communications Library SDK DLL (CSC32.DLL)** can also be used from any language (Visual C++, .NET, Visual Basic, VB.NET, ACCESS, EXCEL, VBA, Delphi, COBOL, Xbase++, Visual FoxPro, etc.) capable of calling the Windows API.

The **Client/Server Communications Programmer's Manual** provides information needed to compile programs using **CSC** in a dBASE 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) and is much smaller than a comparable OCX or ActiveX control.
- Win32 DLL is included.
- **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**), Visual FoxPro (**CSC4FP**), Xbase++ (**CSC4XB**) and Delphi (**CSC4D**). Each version of **CSC** uses the same DLLs (**CSC32.DLL/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 32-bit and 64-bit Windows.
- Supports both UDP and TCP protocols.
- Can be used to create both client and server programs.
- 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.
- Servers can handle multiple connections concurrently.
- Supports secure and private messaging.
- Create chat server and clients.
- Create client / server file transfer programs.
- 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
- 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.
- Specify the maximum number of connections that the server will accept.
- 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.
- Detailed knowledge of Winsock and TCP/IP is not needed.
- Supports Windows XP through Windows 10.
- Works with all versions of 32-bit dBASE (V7.0 through V7.5) and dBASE Plus .
- 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.
- Free unlimited one-year technical support for registered users.

A selection of dBASE example programs with full source code is included. Refer to Section 6 for more details on each of the example projects.

SendUDP	Transmits a UDP data packet to RecvUDP.
RecvUDP	Receives multicast UDP packets from SendUDP.
Cscver	Displays CSC version and build
Client	Simple client example program.
Server	Server example program.
Server2	Server example program (2 concurrent connections).
Download	Downloads text file from HTTP (web) server.
FileGet	File server example that encrypts files.
FilePut	File client example that decrypts files.
Pop3stat	Gets # emails waiting on POP3 server.
Hello	Form that displays CSC version and build.
uNetTime	UDP client gets Network Time.
Control	Sends command to the IP address of a device.

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the first manual (CSC_4DB) in the set.

- [CSC4DB Programmer's Manual](#) (CSC_4DB.PDF)
- [CSC User's Manual](#) (CSC_USR.PDF)
- [CSC Reference Manual](#) (CSC_REF.PDF)

The CSC_4DB Programmer's Manual ([CSC_4DB](#)) is the language specific (dBASE) manual. All language dependent programming issues are discussed in this manual. Information needed to compile programs in a dBASE 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.

All manuals can be viewed online at

<http://www.marshallsoft.com/csc4db.htm>

1.3 Example Program

The following code segment attempts to connect to the server.

```
DataSock = cscClient(HostName, HostPort)
if DataSock < 0
  ? "ERROR: " + Str(DataSock) + " cscClient fails"
  Buffer = Space(128)
  BufLen = cscErrorText(DataSock, Buffer, 128)
  if BufLen > 0
    ? Left(Buffer, BufLen)
  endif
  Code = cscRelease()
  return
endif
? "Connected to server"
```

Also see the example programs in the \CSC4DB\APPS sub-directory where CSC4DB was installed.

1.4 Installation

- (1) Before installation of CSC4DB, Visual dBASE should already be installed.
- (2) Unzip CSC4DB71.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 CSC4DB files. SETUP will also copy CSC32.DLL to the Windows directory. Note that no DLL registration is required.

1.5 Uninstalling

Uninstalling CSC4DB is very easy. First, delete the CSC4FP project directory created when installing CSC4DB. Second, delete CSC32.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 USD (or \$199 USD with source code [ANSI C] to the library DLL). CSC can also be bundled with our **AES Encryption Library (AES4DB)** for strong encryption. 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:

- \$33 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).
- \$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 (CSC)** has been tested on multiple computers running 32-bit and 64-bit Windows XP through Windows 10.

The CSC4DB library will work with all versions of 32-bit Visual dBASE and dBASE Plus. The CSC32.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 DLL to the Windows directory. Refer to Section 1.4 "Installation". After SETUP is run, the CSC4DB files are copied to the directory specified (default \CSC4DB). 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".

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

The keycode is not the customer ID (which is a 4 or 5 digit number).

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 from the Windows search.

2.3 INCLUDE Files

All example programs include two files; `KEYCODE.CC` and `CSC32.CC`. The file `CSC32.CC` contains all the necessary constants and function declarations for `CSC4DB`, while the file `KEYCODE.CC` contains your key code.

There are three recommended ways to handle these `INCLUDE` files in `dBASE` programs.

1. Copy the `INCLUDE` files to the `dBASE` compiler's `INCLUDE` directory.
2. Edit the `INCLUDE` statements (for each program) with their physical location.
3. Replace the `INCLUDE` statements (in each program) by their contents

Visual `dBASE` 7.0 and above, including `dBASE Plus`, create 32-bit applications and use the 32-bit `DLL`, `CSC32.DLL`. Copy the 32-bit `CC` file, `CSC32.CC` to the `dBASE` compiler's `INCLUDE` directory.

2.4 dBASE Forms

`CSC` functions can be called from any Visual `dBASE` code module, such as programs, classes, and forms. See the `HELLO.WFM` example form.

2.5 Dynamic Strings

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

When passing a string buffer to a `DLL` function into which text will be copied, it is strongly recommended that the local string be allocated immediately before use. For example:

```
BufLen = cscErrorText(ErrorCode, Buffer, 128)
if BufLen > 0
    ? Left(Buffer, BufLen)
endif
```

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

2.6 Null Terminated Strings

All strings returned from `CSC` functions are null terminated which means the end of the string is delimited by a `Chr(0)` character. These strings may be converted for `dBASE` in one of two ways: (1) if the length of the string is known, use the `dBASE LEFT` function: For example,

```
* get server IP address
TempBuffer = SPACE(TEMP_SIZE)
Code = fceGetString(0, FCE_GET_SERVER_IP, TempBuffer, TEMP_SIZE)
if Code > 0
    ? "Server IP ", LEFT(TempBuffer, Code)
endif
```

If the length of the null terminated string is not known, use the `FoxPro AT` function to find the position of `Chr(0)`.

2.7 Adding CSC to a dBASE Program

- (1) Add the CSC constants (found in CSC32.CC) that will be used in the developer's application.
- (2) Add the CSC function declarations (found in CSC32.CC) that will be called from the developer's application.

Refer to the example programs.

2.8 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	7.1
BYE	OK (then disconnects)	BYE	OK

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 \CSC4FP\APPS directory.

2.9 Error Display

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

Each sample program contains examples of error processing.

Also see the file cscErrors.txt for a list of all CSC error codes

3 Compiler Issues

The **Client / Server Communications for dBASE (CSC4DB)** component library supports all versions of 32-bit dBASE (Version 7.0, Version 7.5 and dBASE Plus).

CSC4DB does not support 16-bit (Win16) applications.

3.1 Compiling dBASE Programs

dBASE programs end with the extension ".PRG". Before compiling any of the example programs, edit them with your email parameters. Programs can be edited within any text editor, and compiled from the dBASE command window with the COMPILE command (e.g.: COMPILE CONTROL.PRG) or executed from the dBASE command window with the DO command (e.g.: DO CONTROL.PRG).

To open a program within Visual dBASE source editor, choose "File", then "Open". When the "Open File" dialog box appears, choose "Programs" for "Files of Type", then choose the program (*.PRG) to open. Lastly, choose "Open in Source Editor" for "Action" and push the "Open" button.

After editing the source program with your internet (or TCP/IP) parameters, you are ready to compile. From the dBASE menu bar, choose "Build", then "Compile". To run choose, "Run". The dBASE command window must be displayed in order to view the output.

3.2 Compiling dBASE Projects

Visual dBASE projects consist of several types of files such as forms, reports, data modules, etc. The project file itself ends with the extension of ".PRJ".

There is one example dBASE projects (HELLO.). Open HELLO by choosing "File", then "Open Project" from the dBASE menu bar. To compile HELLO, choose "Build" from the menu bar, then "Rebuild All". This will create HELLO.EXE, which can be executed by choosing "Execute hello.exe" from the "Build" menu bar pulldown, or from the Windows command line prompt.

3.3 Creating Executables

dBASE programs end in ".PRG". They can be compiled to an executable using the dBASE BUILD command.

For example, to create CSCVER.EXE from CSCVER.PRG in the C:\CSC4DB\APPS directory, type the following in the dBASE command window:

```
BUILD PROJECT C:\CSC4DB\APPS\WSCVER FROM C:\CSC4DB\APPS\WSCVER
BUILD EXE C:\CSC4DB\APPS\CSCVER FROM C:\CSC4DB\APPS\CSCVER
```

3.4 Compiling CSC Source

The source code for the CSC DLL's is written in standard ANSI C (CSC32.C), and has been compiled using Microsoft Visual C++. The Win32 version is compiled with the STDCALL and DECLSPEC compiler keywords. Source code for the CSC library can be purchased at the same time as a CSC developer license is purchased.

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 dBASE Example Programs

All example programs are written for 32-bit dBASE. Each has been tested and shows how to correctly use CSC functions. It suggested that the developer compile and run the example programs before developing an application using CSC4DB.

Refer to Section 3.1 above for information on compiling and linking the example programs. CSC functions may also be called from Visual dBASE projects.

4.1 CSCVER

The CSCVER ("CSC Version") example program (CSCVER.PRG) displays the CSC version number. This is the first program to compile and build since it verifies that CSC32.DLL is installed properly.

4.2 SendUDP

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

4.3 RecvUDP

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

4.4 Client

The CLIENT example program (CLIENT.PRG) operates as a client that connects to the example server program (SERVER). Edit CLIENT.PRG with your host name or server's IP address before compiling. Start the CLIENT program after the SERVER program.

4.5 Server

The SERVER example program (SERVER.PRG) operates as a server that accepts connections from the example client program (CLIENT). Edit SERVER.PRG with your host name or IP address before compiling.

4.6 FileGet

The FileGet example program (FileGet.PRG) operates as a SERVER, and receives files from the FilePut client. Files are decrypted when received. Edit FileGet.PRG with the host name or server IP address before compiling. Start FileGet before FilePut.

4.7 FilePut

The FilePut example program (FilePut.PRG) operates as a CLIENT, and sends files to the FileGet server. Files are encrypted when sent. Edit FilePut.PRG with the host name or server IP address before compiling.

4.8 Download

The Download example client program (Download.PRG) connects to the MarshallSoft web site (HTTP server) and downloads a file from the `./files` directory.

4.9 POP3Stat

The POP3Stat example client program (POP3Stat.PRG) logs onto a POP3 account and returns the number of emails waiting. Edit POP3Stat.PRG with the POP3 Server name, user name and password before compiling.

4.10 Hello

The Hello example dBASE form displays the CSC version and build number when the command button is pressed.

From the dBASE menu (File/Open Project), open the project HELLO.PRJ. This will load the form HELLO.WFM. From the dBASE BUILD tab, choose "Run hello.wfm" to run the form.

4.11 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`

4.12 Control

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

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

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

5 Revision History

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

Version 6.1: September 23, 2010.

Initial Visual dBASE release of CSC.

Version 6.2: March 9, 2012

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

Version 6.3: July 17, 2013

- Added: New example programs that use AES (Advanced Encryption Standard).
- Fixed: cscCryptoGetFileExt and cscCryptoPutFileExt now uses 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.
- Fixed: cscResolve was sometimes returning bogus IP addresses after the first.

Version 7.0: April 6, 2015

- Added cscMulticast() that receives multicast UDP packets.
- Added cscClientExt() that binds to a local IP address (for multi-homed computers)
- Fixed bogus CSC_BAD_OFFSET error sometimes returned by cscCryptoPutFile().

Version 7.1: January 26, 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_SOCKET_REUSE [to cscSetInteger], enabling an app to close the listening socket and immediately reopen without error.