Client / Server Communications

Library for Xbase++

Programmer's Manual

(CSC4XB)

Version 7.1

February 2, 2018

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

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

Web: www.marshallsoft.com

TABLE OF CONTENTS

1 Introduction

The **Client / Server Communications Library for Xbase++ (CSC4XB)** is a toolkit that allows software developers to quickly create server and client TCP/IP and UDP applications in Alaska Xbase++

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 Xbase++ (CSC4XB)** component library supports and has been tested with Alaska Xbase++ version v1.3 through Xbase++ v1.9. **CSC4XB** includes several Xbase++ 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 the MarshallSoft AES Advanced Encryption Library (**AES4XB**) if strong encryption is desired.

CSC4XB runs under all 32-bit and 64-bit versions of Windows through Windows 10. The **Client / Server Communications Library SDK** DLL (CSC32.DLL/CSC64.DLL) can also be used from any language (Visual C++, .NET, Visual Basic, VB.NET,ACCESS, EXCEL, VBA, Delphi, COBOL, Visual dBase, etc.) capable of calling the Windows API.

The **Client/Server Communications Programmer's Manual** provides information needed to compile programs using CSC in an Xbase++ 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), Delphi (CSC4D), Visual FoxPro (CSC4FP), dBase (CSC4DB), and Xbase++ (CSC4XB). Each version of CSC uses the same DLL (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 evaluation version is available. Contact us if you have any questions.

1.1 Features

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

- Supports both 32-bit and 64-bit Windows.
- Supports both UDP and TCP protocols.
- Can be used to create both client and server programs.
- Supports "one time" passwords for improved security.
- Can encrypt/decrypt data and files being transmitted.
- Use with the 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.
- 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.
- Create client programs to talk to TCP servers (POP3, IMAP, HTTP, SMTP, DNS)
- Create SMTP proxy programs extracting a copy of all recipient addresses
- Create POP3 proxy programs that filter incoming email for Spam
- Create HTTP proxy used to filter content
- Specify the maximum number of connections that the server will accept.
- Allows multiple servers and clients to run simultaneously.
- 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 32-bit and 64-bit windows from Windows XP through Windows 10.
- Works with all versions of 32-bit Xbase++, from V1.3 through V1.9.
- 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.
- Documentation online as well as in printable format.

A selection of Xbase++ 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.
Client        Simple client example program.
Control       Sends command to the IP address of a device.
cscVer        Displays CSC version and build
Download      Downloads text file from HTTP (web) server.
FileGet       File server example that encrypts files.
FilePut       File client example that decrypts files.
Form          Form that displays CSC version and build.
Pop3Stat      Gets # emails waiting on POP3 server.
Server        Server example program.
uNetTime      UDP client gets Network Time.
```

1.2 Documentation Set

The complete set of documentation consists of three manuals. This is the first manual (CSC_4XB) in the set.

- CSC4XB Programmer's Manual (CSC_4XB.PDF)
- CSC User's Manual (CSC_USR.PDF)
- CSC Reference Manual (CSC_REF.PDF)

The CSC_4XB Programmer's Manual (CSC_4XB) is the language specific (Xbase++) manual. All language dependent programming issues are discussed in this manual. Information needed to compile programs in an Xbase++ environment is provided in this manual. Examples programs are also discussed.

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/csc4xb.htm

1.3 Example Program

The following code segment attempts to connect to the server.

```
* attach DLL (allocating 1 data socket)
if XcscAttach(1, 0, CSC_KEY_CODE) < 0
  ?"ERROR: Bad Key Code!"
  return
endif
* get version & build
nVersion = XcscGetInteger(-1, CSC_GET_VERSION)
* compute CSC version
A = int(nVersion / 256)
nVersion = nVersion - (256 * A)
B = int(nVersion / 16)
C = nVersion - (16 * B)
? "CSC Ver "+LTRIM(Str(A))+"."+LTRIM(Str(B))+"."+LTRIM(Str(C))
* get build
nBuild = XcscGetInteger(-1, CSC_GET_BUILD)
? "CSC Bld " + Ltrim(Str(nBuild))
return
```

Also see the example programs in the \CSC4XB\APPS sub-directory where CSC4XB was installed.

1.4 Installation

(1) Before installation of CSC4XB, Xbase++ should already be installed.

(2) Unzip CSC4XB71.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 CSC4XB files. SETUP will also copy CSC32.DLL to the Windows directory. Note that no DLL registration is required.

1.5 Uninstalling

Uninstalling CSC4XB is very easy. First, delete the CSC4XB project directory created when installing CSC4XB. 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). Purchasing details can be found in Section 1.4, "How to Purchase", of the CSC User's Manual (CSC_USR).

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

1.7 Updates

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

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

After one year, the developer license must be updated to be able to download updates. The license can be updated for:

- $33 if the update is ordered within one year of the original purchase (or previous update).
- $55 is the update is ordered between 1 and 3 years of the original purchase (or previous update).
- $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 Windows XP through Windows 10.

The CSC4XB library will work with all versions of 32-bit Xbase++. 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. CSC64.DLL is available to use with Win64 applications.

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 CSC4XB files are copied to the directory specified (default \CSC4XB). 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.CH. The keycode for the evaluation version is 0. You will receive a new key code when registering and a new CSC32.DLL. 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 Win32 STDCALL and DECLSPEC

CSC32 is written in ANSI C and is compiled using the _stdcall and _declspec keywords. This means that CSC32 uses the same calling conventions and file naming conventions as the Win32 API. In particular, function names are NOT decorated. There are no leading underscores or trailing "@size" strings added to function names.

The CSC32.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 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:

REQ	Response String	Request Example	Response Example
WHO	Sends name of the server.	WHO	SERVER
VER	Sends server version #.	VER	7.1
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 \CSC4XB\APPS directory.

2.5 Adding CSC4XB to a Project

It is straightforward to add **CSC** to Xbase++ programs. First, add

```
#INCLUDE "KEYCODE.CH"
#INCLUDE "CSC32.CH"
```

after any other `$INCLUDE` statements in the Xbase++ program.

Then add

```
nCode = cscAttach(1 CSC_KEY_CODE)
If nCode < 0 Then
  ? "Cannot attach CSC"
  return
endif
```

as the first executed **CSC** function.

The keycode (contained in KEYCODE.CH) is 0 for the evaluation version and is a 9-digit number for the purchased version. Rather than include KEYCODE.CH as shown above, the keycode can be pasted directly into the call to **seeAttach**.

2.6 INCLUDE Files

All example programs include two files: `KEYCODE.CH` and `CSC32.CH`. The file `CSC32.CH` contains all the necessary constants and function declarations for CSC4XB, while the file `KEYCODE.CH` contains your keycode (license key), as discussed in Section 2.2.

The Alaska Xbase++ include file DLL.CH is also required. For example,

```
#INCLUDE "DLL.CH"
#INCLUDE "KEYCODE.CH"
#INCLUDE "CSC32.CH"
```

The above files can be copied to the Xbase++ INCLUDE directory (where Xbase++ can find it) if so desired.

Note that each function is declared with the prefix character of 'X'

2.7 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

CSC4XB has been compiled and tested with Alaska Xbase++ versions v1.3 through and Xbase++ v1.9. The SETUP installation program will copy the Lib's and CSC32.DLL to the Windows directory. Refer to Section 1.4 "Installation".

3.1 Compiling and Linking Programs

Before compiling any of the example programs, edit each program with your TCP/IP parameters, as shown in the example program in Section 1.3 above. Server names can be IP addresses (in decimal dot notation) or the host name.

More details on each of the example programs are provided in Section 4.0, "Example Programs.

To compile and link console mode programs, such as `SERVER.PRG`, use

```
xpp server.prg
alink /subsystem:console server.obj
```

To compile and link windows GUI programs, such as `FORM.PRG`, use

```
XPP xpp form.prg
alink /subsystem:windows form.objalink /subsystem:windows form.obj
```

3.2 Xbase++ Compiler

If you don't have the Alaska Software Xbase++ compiler, you can find it on the web at

```
http://www.alaska-software.com
```

3.3 Compiling CSC Source

The source code for the CSC DLL 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.

CSC may also be compiled using Borland C/C++ or Watcom C/C++ compilers. If you recompile CSC32.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 Xbase++ Example Programs

Edit programs, such as CLIENT.PRG, with your TCP/IP email parameters before compiling

Before writing your own programs, compile and run several of the example programs.
Refer to Section 3.2 above for information on compiling and linking the example programs.

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 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.

4.3 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.

4.4 Download

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

4.5 Form

The Form example form program (FORM.PRG) displays the CSC version and build number when the command button is pressed.

4.6 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

4.7 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.

4.8 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.

4.9 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.

4.10 FileGetX

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

FileGetX uses the cscGetFileExt function (see section 2.10 in CSC Users Manual).

4.11 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. Start FileGet (the server) before FilePut.

4.12 FilePutX

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

FilePutX uses the cscPutFileExt function (see section 2.10 in CSC Users Manual).

4.13 SendUDP

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

4.14 RecvUDP

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

5 Revision History

CSC32.DLL is written in ANSI C. All language versions of CSC (C/C++, Visual Basic, Delphi, Visual FoxPro, and Xbase++) use the same DLL.

Version 6.0: August 26, 2009

Initial Xbase++ release of CSC.

Version 6.1: October 7, 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: cscMakeDotted4() function (constructs dotted IP address from components).
- Added: Control.prg example program (direct TCP control of external relay).

Version 6.2: February 29, 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: June 20, 2013

- Added new example programs that use AES (Advanced Encryption Standard).
- Fixed: cscCryptoGetFileExt & cscCryptoPutFileExt not using local file path.
- Changed cscFillRandom(*,*,0) to use random seed rather than passed seed.
- cscSetInteger (Port, CSC_SET_CLOSE_TIMEOUT, Tics) sets max ticks before socket is forced closed.

Version 7.0: April 7, 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 crcCryptoPutFile().

Version 7.1: February 2, 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.