SecureTCPServerExample

README

Last Updated August 8th, 2019

Contents

[Overview 3](#_Toc16147708)

[Console Commands 3](#_Toc16147709)

[Listen 3](#_Toc16147710)

[Disconnect 4](#_Toc16147711)

[ShowStatus 4](#_Toc16147712)

[Equipment 5](#_Toc16147713)

[Devices 5](#_Toc16147714)

[Software / Firmware 5](#_Toc16147715)

[Important Notes 5](#_Toc16147716)

[Before Loading the Program 5](#_Toc16147717)

[Loading the Program and Project Files 5](#_Toc16147718)

# Overview

This SecureTCPServerExample project was created to exercise the SecureTCPServer class from the SIMPL# Pro library. It provides a console command interface through which users can set up a SecureTCPServer and manage up to three client connections asynchronously. The server itself is an "echo server;" once a connection has been established with a client, any text sent from client to server will be echoed back to that client.

Users can experiment with the SecureTCPServerExample project using either an OpenSSL s\_client, the corresponding SecureTCPClientExample project located at crestron.com Support -> Technical Resources -> Example Programs, or some equivalent TLS1.2 client.

**Note:** The SecureTCPServer does not perform client authentication.

All SIMPL# Pro source code defining the program is provided in the following files:

* ControlSystem.cs
* SecureServerTest.cs

These source files are thoroughly commented and are intended to provide an illustrative application of various SIMPL# Pro APIs.

## Console Commands

The SecureTCPServerExample program provides three console commands; they allow the user to begin listening for clients on a particular port, to disconnect one (or all) clients, and to show the current status of a particular client connection as well as the server's current state (listening/not listening, has connections/does not have connections):

### Listen

*listen [<cert\_file> <key\_file>] <port>*

You must provide the port number for the SecureTCPServer to begin listening on, and you may optionally provide a digital certificate and its corresponding private key file to be associated with the server. If you choose to provide a certificate and key, you must store them both in DER format into the Application directory of the program, then provide their filenames as inputs to the console command.

To do this in Visual Studio 2008, right click on the solution name, "SecureServerExample," in the Solution Explorer pane, and select Add an Existing Item. Navigate to your certificate and key files and add them to the solution. Next, click on the files in the Solution Explorer and change the "Copy To Output Directory" option to "Copy always" so that they appear in the application directory when you load the program onto the control system.

If no user certificate and key are provided, the SecureTCPServer will use the SSL state of the control system to determine which certificate to use. That is, SSL Self will use the self-signed certificate of the control system.

**Note:** SSL must be enabled on the controller, *unless* you are providing your own certificate and key via the **listen** console command.

### Disconnect

*disconnect [<client\_index>]*

Optionally, you can provide a client index, an integer representing the client connection, to close that particular connection if it is currently open. The client index will appear in the console whenever a new client connects.

If you instead provide no argument to Disconnect, the client will disconnect from all currently connected clients and it will stop the server altogether. At this point, you can invoke the Listen console command to start the server up again, possibly with a new port number or a new certificate/key configuration if you want.

### ShowStatus

*showstatus [<client\_index>]*

Provide a client index, an integer representing the client connection, to print the current status of the server's connection with this client. The connection status will be a string representation of the SocketStatus enumeration, which can take on possible values such as SOCKET\_STATUS\_CONNECTED, SOCKET\_STATUS\_NO\_CONNECT, or SOCKET\_STATUS\_WAITING. This can be used to keep track of which clients are currently connected to the server, as up to 3 simultaneous connections are possible by default and the maximum can be reconfigured by editing the max\_connections local variable in the SecureServer.Listen method. In addition, this console command will always print the current server state, taken from the State property of the internal SecureTCPServer object.

If no argument is provided to ShowStatus, this console command will just show the current server state, as represented by the ServerState enumeration containing 3 distinct flags: SERVER\_LISTENING, SERVER\_NOT\_LISTENING, and SERVER\_CONNECTED.

For example, a completely disconnected server would be in the state "SERVER\_NOT\_LISTENING," while a server at maximum connection capacity would have the state "SERVER\_NOT\_LISTENING, SERVER\_CONNECTED." A server with some active connections that also has room for more would be in the "SERVER\_LISTENING, SERVER\_CONNECTED" state, while a listening server that hasn't received any clients yet would simply be in the "SERVER\_LISTENING" state.

# Equipment

This program is designed to work with the following hardware/software:

## Devices

* 3-Series Control System
* One or more separate TCP clients supporting TLS1.2, such as an OpenSSL s\_client running on a laptop or the SecureTCPClientExample program running on another control system

## Software / Firmware

|  |  |
| --- | --- |
| Device | Firmware Version |
| 3-Series Control System | 1.601.xxx or later |

|  |  |
| --- | --- |
| Software | Version |
| Toolbox | 3.03.xxx or later |

# Important Notes

## Before Loading the Program

* Set SSL to SELF (or CA if possible). If you want to associate your own digital certificate and key with the SecureTCPServer, prepare those two resources as files in DER format (See Overview -> Console Commands -> Listen for more information).

# Loading the Program and Project Files

1. Open the top-level solution file, SecureServerExample.sln, in Visual Studio 2008.
2. Press F6 to Build All. This will place the .cpz file in the SecureServerExample\bin\Debug folder.
3. Transfer the .cpz file to one of the control system’s program slots.
4. On the control system, invoke **progload -p:#**, where **#** is the number of the slot into which you placed the .cpz file.