SecureTCPClientExample

README

Last Updated August 8th, 2019

Contents

[Overview 3](#_Toc16153042)

[Console Commands 3](#_Toc16153043)

[Connect 3](#_Toc16153044)

[Send 4](#_Toc16153045)

[Disconnect 4](#_Toc16153046)

[ShowStatus 4](#_Toc16153047)

[Equipment 5](#_Toc16153048)

[Devices 5](#_Toc16153049)

[Software / Firmware 5](#_Toc16153050)

[Important Notes 5](#_Toc16153051)

[Loading the Program and Project Files 5](#_Toc16153052)

# Overview

This SecureTCPClientExample project was created to exercise the SecureTCPClient class from the SIMPL# Pro library. It provides a console command interface through which users can set up a SecureTCPClient and connect to a remote server. Once connected, users may invoke the **send** console command to asynchronously transmit a string of text to the server; the client will also print any text sent its way to the console.

Users can experiment with the SecureTCPClientExample project using either an OpenSSL s\_server, the corresponding SecureTCPServerExample project located in the SIMPL# Pro Examples repository, or some equivalent TLS1.2 server.

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

* ControlSystem.cs
* SecureClientTest.cs

These source files are thoroughly commented and are intended to demonstrate one of many possible applications of the SIMPL# Pro APIs.

## Console Commands

The SecureTCPClientExample program provides four console commands; they allow the user to connect to a server using TLS1.2, send a string of ASCII text to the server, disconnect from the server, and show the current socket status of the client.

### Connect

*connect [<cert\_file> <key\_file>] <hostname> <port>*

The <hostname> and <port> parameters here refer to the hostname and port of the remote server that the client will connect to. You may optionally provide a digital certificate and its corresponding private key file to be associated with the client for the purpose of client authentication. 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, "SecureClientExample," 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.

### Send

*send <msg>*

Send a string of text to the server with which the client is currently connected. If the client is not currently connected to a server, invoking this console command will yield an error response.

### Disconnect

*disconnect*

Disconnect the SecureTCPClient from the server with which it is currently connected. If the client is already disconnected, this console command has no effect.

**Note:** If you intend to switch to a different server, you must invoke this console command first, before the subsequent call to **connect**. This properly ends the current TCP connection and returns the program to a state where it is ready to connect to a new server.

### ShowStatus

*showstatus*

If a SecureTCPClient object has been instantiated with the **connect** console command, **showstatus** will indicate the current socket status of the client as well as the hostname and port of the remote server it is currently configured to connect with.

This console command takes no arguments and simply prints the current socket status of the SecureTCPClient. The connection status will be a string representation of the SocketStatus enumeration, which has values such as SOCKET\_STATUS\_CONNECTED, SOCKET\_STATUS\_NO\_CONNECT, or SOCKET\_STATUS\_WAITING. This command could be used to troubleshoot unexpected closures of the secure TCP connection.

# Equipment

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

## Devices

* 3-Series Control System
* A separate TCP server supporting TLS1.2, such as an OpenSSL s\_server running on a laptop or the SecureTCPServerExample 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:**

* If you want to associate your own digital certificate and key with the SecureTCPClient for the purpose of client authentication, prepare those two resources as files in DER format (See Overview -> Console Commands -> Connect for more information).

# Loading the Program and Project Files

1. Open the top-level solution file, SecureClientExample.sln, in Visual Studio 2008.
2. Press F6 to Build All. This will place the .cpz file in the SecureClientExample\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.