
SuperCom

TCP/IP sample project client/server

ServerClients

© ADONTEC, 2007-2010. All Rights Reserved.

www.adontec.com

ADONTEC®

This document describes the sample project Server/Clients for *SuperCom for TCP/IP* and *SuperCom Suite*.

Technical Support

You can reach ADONTEC technical support by email at support@adontec.com or most secure, via <http://adontec.com> | Support.

Copyright Notice

© ADONTEC, 2007-2010. All Rights Reserved.

This documentation contains proprietary information of ADONTEC and distribution is limited to authorized licensees of ADONTEC. Any unauthorized reproduction or distribution of this document is strictly prohibited.

The Project ServerClients

This project combines a TCP/IP Server and two TCP/IP Clients.

The software is a stripped-down real world high speed, high load, high performance client/server solution. In the current release the server application is accepting concurrent connections, expects clients to login and drop one or more data packets.

Concurrent connections and high performance

You can read a lot about Windows weakness regarding high performance TCP/IP server and concurrent connections. This is by far overstated and may be true in case one uses the wrong tools. *SuperCom* software does prove the opposite.

The current release of this project was optimized, tested and debugged using different desktop PC running Windows XP SP3, Windows XP x64, Windows Vista, Windows 2008 Server and Windows 7. Some tests additionally occurred within virtual machines in order to test the behavior of the software under virtual conditions. Virtual PC 2007 v6 was used having Windows XP SP2 and Windows 7 running in each VM.

CPU load

The server application was showing about 2% to 15% CPU, while having to handle about 100 concurrent connections and each client about 1%-7% running 5 to 50 concurrent connections. This specific measurement performed in the lab on different desktop PC running AMD Athlon 64 X2 DUAL at 2.51GHz with 1GB RAM and Windows XP SP3. Even the CPU values are related in some degree to the used CPU speed, the Windows version, the actual network traffic,... it shows clearly how effective the software combination is.

Customization

Both applications, Server and Client, are developed in functional modules. Only the affected files need to be changed. This makes customization easier.

The important parts and functions are also documented and explained.

Words from the authors

As with all multi threaded applications, implement changes with care and always take into consideration that things may happen really fast and concurrently. If you're unfamiliar with multi threading, take your time to read about it – make small samples and test critical sections and concurrency.

Take any steps to ensure that shared resources are accessed serialized.

Take any precautions to protect shared resources from concurrent use.

Always be prepared that many threads may access the same code or data concurrently. Try to have the data accessed by the threads as separated as possible in order to avoid concurrency.

... and finally test, test and test – many hours and days, before you ship it to your customers. Let it run for many hours and days without debug code or traces.

That's the burden one has to- and should -take in order to get any application up and running error free and especially this kind of.

The Server

The server compiles to the executable *TstSrvClients.exe*.

This server performs a multi threaded server. Once started, it expects one or more clients to connect, login and drop one or more data packets.

The used protocol is simple and explained inside the source code. The login procedure is optional and can be disabled, if needed, by a single define. The server also includes functionality to compile as a Windows Service.

If not otherwise configured in the file *config.ini*, the server will listen to "host:9000".

Server Configuration File

The configuration file CONFIG.INI is expected to be in the same directory as the TstSrvClients.exe file.

It provides a way to set the ip, port and login information.

It also enables to trace different ways.

```
[CONFIG]
;-- Server configuration
IP=host
PORT=9000
;-- Login information
USER=server_client
PASSWORD=admin

[TRACE]
TraceDebug=0
;-- Route outputs to a file
TraceToFile=0
;-- Route outputs to a console window
TraceToConsole=0
;-- Tracing SuperCom data on low level incl. Display format Hex, Dec, Mixed)
TraceCom=0
TraceHex=0
TraceMix=0
```

SuperCom Configuration File

The configuration file SUPERCOM.INI is expected to be in the same directory as the SuperCom.dll file.

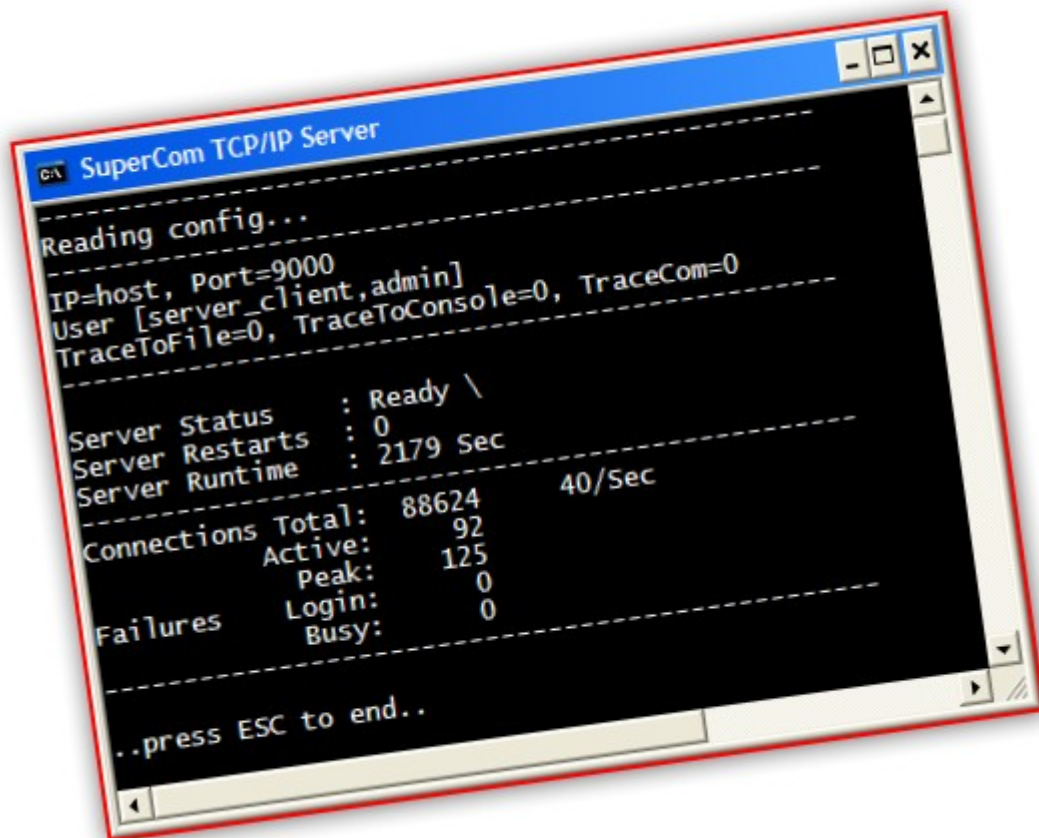
It provides a way to configure SuperCom library with defaults.

```
[GLOBAL]
; Server should try to queue up to 255 concurrent connections
BACKLOG=255
; Since we use small packets and a lot of threads spare on used heap space.
; If any link needs more, it can additionally call ComSetBufSize and increase it.
DEF_TX_BUFFER=1024
DEF_RX_BUFFER=1024
```

Starting the Server

Start as console application (standard way)

```
TstSrvClients -CON
```



The screenshot shows a Windows console window titled "SuperCom TCP/IP Server". The output text is as follows:

```
Reading config...
-----
IP=host, Port=9000
User [server_client,admin]
TraceToFile=0, TraceToConsole=0, TraceCom=0
-----
Server Status      : Ready \
Server Restarts   : 0
Server Runtime    : 2179 Sec
-----
Connections Total: 88624      40/Sec
Active:          92
Peak:           125
Failures Login:  0
Busy:           0
-----
..press ESC to end..
```

NOTE

If your *Firewall* intervenes let it pass!

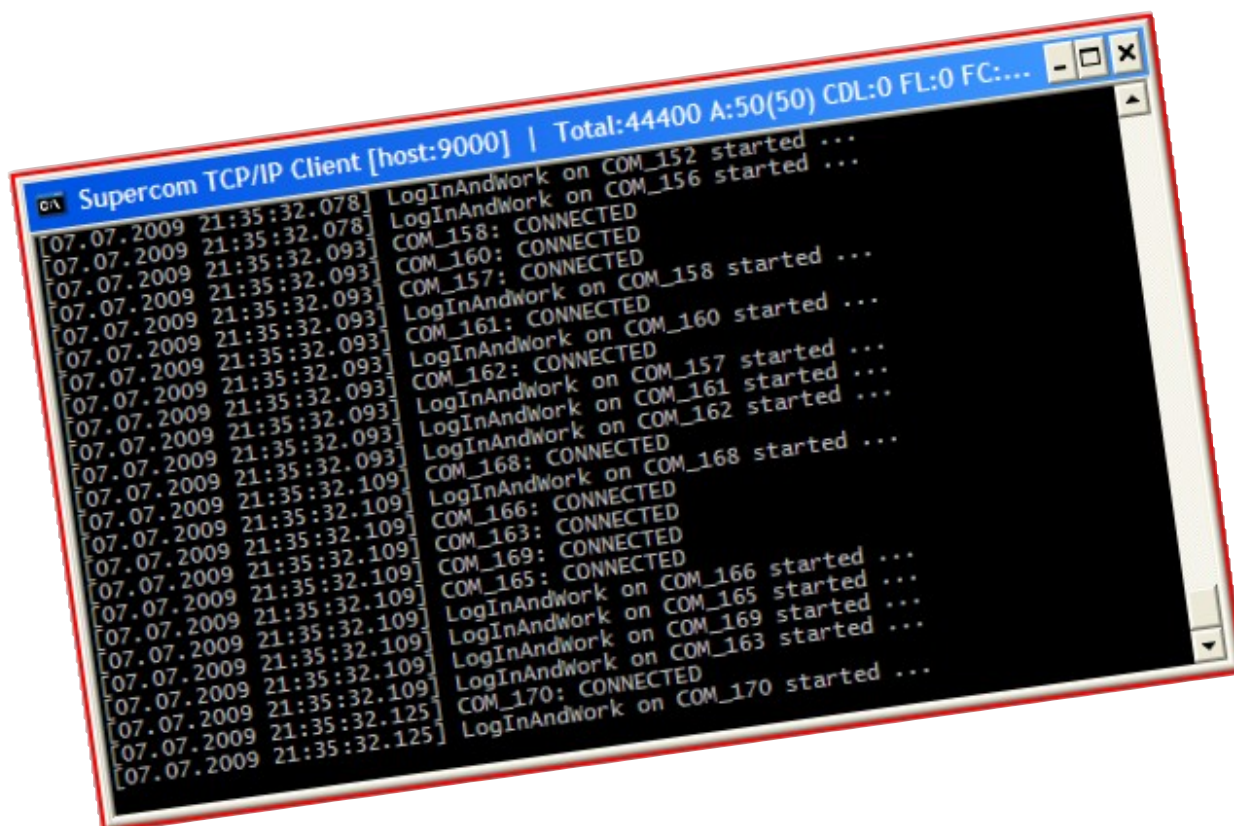
Client.exe - The Client

Once the server started, you can connect with this client.

```
client -axxx
```

where xxx is the server's IP address and PORT address e.g. "host:9000". If the port omitted, the default 9000 is taken.

e.g. `client -ahost:9000`



```
client -ahost -L
```

will repeatedly call the server on the local machine.

NOTE

Once started press any key to perform. If your *Firewall* intervenes let it pass!

Compile

Both sample require the files `supercom.h` and `supercom.lib` to compile.

In order to compile as a Windows Service, a license of the library *ServiceLib* is required.

Server as Windows Service

The server may also run as Windows Service, if it was compiled as a service.

In order to compile as a service, the library "ServiceLib" is required. The source code is already prepared.

Copy the files `adservice.h` and `adservice.lib` from the product "ServiceLib" to the servers project directory and the `adservice.dll` to the "COMMON" subdirectory and re-compile.

Install server as a service

```
TstSrvClients -r
```

Remove server from a service

```
TstSrvClients -d
```

NOTE:

A license of *ServiceLib* is not required if you don't compile a service.

FAQ

What is "host" ?

It's a *SuperCom* specific extension like "localhost" instructing *SuperCom* to query and use the local IP. If the local IP is not known, use „host“ and not „localhost“. The „localhost“ may result to the IP 127.0.0.1, which is the local loop back and packets do not get routed. Tests are more realistic, if packets are routed.

In short: „host“ = the local IP.

You may also use the "Computer Name" configured in Windows e.g. "HPSF01" as alias for the IP but you must query for it. Using „host“ you let *SuperCom* do its job and you're done.

A client located on a remote PC must use the IP address of the server (e.g. 192.168.0.122) or some of its alias names (e.g. "HPSF01" etc.) and the applications Port address, in order to connect e.g. "192.168.0.122:9000". If server and client are located on the same computer using „host“, it makes things easier.

What is a communication channel index ?

SuperCom is controlling each connection using a *Communication Channel Index* (see parameter *Com*). This is nothing more than a real index to internal data structures and counts from 0=COM_1 to 254=COM_255. Thus „COM_1“ is referring to the first *SuperCom* internal communication channel and COM_255 (LAST_COM) to the last one.

The *Communication Channel Index* maps statically to a serial device name only if used with serial devices ie. COM_1 maps to „COM1“ if used on a serial device (see *ComType*).

How many connections does the server support ?

A server can handle up to 254 connections simultaneously.

This is standard for the *SuperCom* software as shipped with factory defaults.

Other restrictions may also apply due to Windows limits on shared resources.

How many connections can a single client establish ?

A client can handle up to 255 connections simultaneously. If you need more just start the same client again.

This is standard for the *SuperCom* software as shipped with factory defaults.

Other restrictions may also apply due to Window's limits on shared resources.

Which compiler to use ?

Both, client and server applications, were created and are maintained for efficiency with Visual C++ 6. They compile with Visual Studio C++ 2003, 2005, 2008, with C++ Builder as well as with other C/C++ suites. The *SuperCom* header files are compatible with many of the very known compiler suites.

The *bcbClient* was created and maintained with C++ Builder.