

## **COBOL sample application: multithreaded COBOL socket server with single-threaded COBOL client**

Please see also Knowledge Base article [23143](#).

This is a sample application which may be of interest. The program server.cbl is the socket's server ( written in COBOL ). The program client.cbl is the socket's client ( written in COBOL ).

This sample was mainly tested on AIX 5.2.0.0 with Server Express 4.0 SP2 32bits.

To Compile and link the sample

=====

To compile and link server.cbl & client.cbl, launch cl.sh

cl.sh:

```
cob -xagt server.cbl
```

```
cob -xag client.cbl
```

```
cob -i launchnclient.cbl
```

```
cob -i clientStopServer.cbl
```

To run the sample

=====

To run the sample, open 2 UNIX sessions.

Login as root in the first session... this session will launch the socket's server,  
the program named server

Login with a 'normal' UserId in the second session... this session will launch the  
socket's client,

the program named client

socket's server Command Line

=====

The command-line of the program server is: Usage: server socket-port-number

If, at the prompt,

you type: server or server 0, a socket's port-number of 903 will be used

you type: server 903

the sockets API getservbyname will be used to find the service in etc/services  
(The getservbyname subroutine retrieves an entry from the /etc/services file using  
the service name as a search key)  
!!!! need to add, in etc/services, CobolSocket 903/tcp

you type: server nnn ( nnn not being equal to 0 or 903 )  
the sockets API getservbyport will be used to find the service in etc/services  
(The getservbyport subroutine retrieves an entry from the /etc/services file using a  
port number as a search key)  
!!!! need to add, in etc/services, CobolSocket 903/tcp

!!!! The simplest is just to type: server

#### socket's client Command Line

=====

The command-line of the program client is: client ServerNameOrAddress port-number

If, at the prompt,

you type: client,

the client program will 'search' the socket's server on the port 903 of the current  
machine

you type: client aHostName

the client program will 'search' the socket's server on the port 903 of the machine  
named aHostName

you type: client aHostName 903

sockets API getservbyname will be used to find the service in etc/services  
(The getservbyname subroutine retrieves an entry from the /etc/services file using  
the service name as a search key)

!!!! need to add, in etc/services, CobolSocket 903/tcp

you type: client aHostName nnn ( nnn not being equal to 0 or 903 )

sockets API getservbyport will be used to find the service in etc/services  
(The getservbyport subroutine retrieves an entry from the /etc/services file using a  
port number as a search key)

!!!! need to add, in etc/services, CobolSocket 903/tcp

!!!! The simplest approach is just to type: client

client )  
or cobrun launchnclient ( which creates 50 processes launchling

!!!! To stop the socket server, from the client session, launch `cobrun clientStopServer`

demonstration:

=====

The code is contained in [multithread\\_socket.tar](#)

-----

A 'Message Header' has been included in messages sent on the socket so the socket's client and socket's server can control when it is normal they received this message

This 'Message header' contains the Server socket's identifier AND the client Socket's identifier AND a message number

Note, if the demonstration, constants are defined.

They allow:

mode ) to display or not the content of the message header ( `display == debug`

( `constant DebugMode-MsgHeader` )

and client to display or not the content of messages sent and received by the server

( `constant DispMsgRecvSent` )

>>>> Comment or set these constants depending on the need  
( 'debug mode' or 'run mode' )

Addendums:

=====

The source code contains the URLs leading to descriptions of the sockets APIs.

Another interesting URL:

<http://www.cs.rpi.edu/courses/sysprog/sockets/sock.html>

Sockets Tutorial