```
`tcp_simpleServer
`very basic and simple TCP server for text data
`needs to be run from within it's own 4D process
`$vl_processID:=New process("tcp_simpleServer"; 64*1024;"tcp_simpleServer")

`you need to launch the server before the 4D simple tcp client can be

`this simpleTCP sever is setup to deal with one single incoming  connection at a time
`before allowing a new ip address to connect to it or to allow for  many ip sockets to be
`hitting it all at the same time

`the intention of this 4D simpeTCP server is to show you how a tcp  connection works between the 4D simpleTCP client method
`it does not deal with all possible issues rather it is meant to  help you learn how to make 4D TCP calls to+from 4D

`I would also suggest you place a trace in to step through the  code to see it in action

$vl_error:=IT_MacTCPInit   `you need to turn on mac tcp driver for 4D  IC commands put in db startup si tonly needs to be done once



xvb_weStopTcpServer:=False

$vl_remotePort:=7777  `set to what port number you want to use

Repeat

   $vt_inComingData:=""

   $vl_error:=TCP_Listen ("";0;$vl_remotePort;$vl_timeOut;$vl_socket)
   $vl_error:=TCP_State ($vl_socket;$vl_Status)

   If ($vl_Status=8)  `connection established
```

```
$vl_timeOut:=Tickcount+(60*3)  `3 seconds adjust to what ever you  need or want

Repeat

   $vl_error:=TCP_Receive ($vl_socket;$vt_buffer)  `get size of  incoming text buffer to look for
   $vl_size:=Num($vt_buffer)

Until (($vl_size>0) | ($vl_error#0) | (Tickcount>$vl_timeOut))

$vt_buffer:=""


 `trace  `if you want to see the server in debug mode if you do  change the timeout period to longer tickcounts

If ($vl_size>0)  `we have a doc size to look for

   $vl_timeOut:=Tickcount+(60*10)  `10 second timeout adjust to what  ever you want or need it to be

  Repeat

     $vl_error:=TCP_Receive ($vl_socket;$vt_buffer)  `not all  data  may show in one hit sit in loop to you get correct size
     $vt_inComingData:=$vt_inComingData+$vt_buffer

     $vl_error:=TCP_State ($vl_socket;$vl_State)  `0 = closed   2=  established  8=connected

  Until (($vl_State=0) | ($vl_error#0) | (Tickcount>$vl_timeOut) |
  (Length($vt_inComingData)=$vl_size))

   $vl_error:=TCP_Send ($vl_socket;"OK")
   $vl_error:=TCP_Close ($vl_socket)

     `do what you need to with the data   here------------------------------
```

```
            `for now we will post the text into a 4D dialog that sit up for  1 second
            $vl_left:=400
            $vl_top:=250
            $vl_width:=500
            $vl_height:=250

            Open window($vl_left;$vl_top;$vl_left+$vl_width;$vl_top+$vl_height;5;"simple tcp server said")  ` take this out as it's here jus
            MESSAGE($vt_inComingData)
            DELAY PROCESS(Current process;60*1)  `wait for3 seconds
            CLOSE WINDOW

            `make sure during this test period that the user has allowed   this dialog to close before sending a new message
            `to this server as it is currently set up to deal with only one   incoming connection at a time
            `to deal with more than 1 message you will need to have a thread   pool to set up a master listening server that
            `will pass off incoming conections to the thread pool to deal  with them in their own slave receiver process


            ` you could also turn ?vb_weStop to true here if you want to  kick out of main loop now

         End if

      End if

   Until (xvb_weStopTcpServer)

   ALERT("the simpleTCP server will shut down now")

   `eom - tcp_simpleServer
```

```
` basic tcp client
`very simple + very basic tcp client for text date
`should be run from it's own 4D process
`$vl_processID:=New process("tcp_simpleClient"; 64*1024;"tcp_simpleClient")

$vl_remotePort:=7777  `set to what port number you want to use
$vt_remoteAddress:="192.168.0.1"  `set to your connecting ip address  or use the command net_resolve to change a hosts name to it's

xvb_weStopTcpClient:=False  `place in db startup or as needed



C_TEXT($vt_buffer)

Repeat

   $vt_Send:=Request("type a text message to send here")  `do not send a second message before the simpleTCP server closes the tes
    `   see simpleTCP server for more info

  If (OK=1)  `good to go we have some text to send now

     $vl_err:=TCP_Open ($vt_remoteAddress;$vl_remotePort;$vl_socket)

    If ($vl_socket>0)

       $vl_err:=TCP_Send ($vl_socket;String(Length($vt_Send)))  `tell the  server how much data you will be sending here in a mom

       $vl_restTime:=Tickcount+(60*1)  `wait for a timeperiod to make  sure the server has recieved the ok message before sendir
      Repeat
         IDLE
      Until (Tickcount>$vl_restTime)

       $vl_err:=TCP_Send ($vl_socket;$vt_Send)  `send your data. This  could be changed to TCP_SendBlob but you will have to do t
```

```
        $vl_timedOut:=Tickcount+(60*5)  `wait x seconds -- if no message  then time out can break out of receive loop
      Repeat
         $vl_err:=TCP_Receive ($vl_socket;$buffer)
         $vl_err:=TCP_State ($vl_socket;$vl_State)
         $vt_buffer:=$vt_buffer+$vt_buffer

      Until (($vl_State=0) | ($vl_err#0) | (Tickcount>$vl_timedOut))  `  until host closes connection or an error--could also add a ti

        `deal with buffer message or connection error code here

    Else

      MESSAGE("we could not connect to server -- please check your settings")

    End if

  Else

    xvb_weStopTcpClient:=True  `this set here for testing purpose only  take the request dialog out to really use this code for someth
    xvb_weStopTcpServer:=True
    ALERT("the simpleTCP client + server will start the shut down  process now")



  End if



Until (xvb_weStopTcpClient)
```

```
`eom - basic tcp client
```