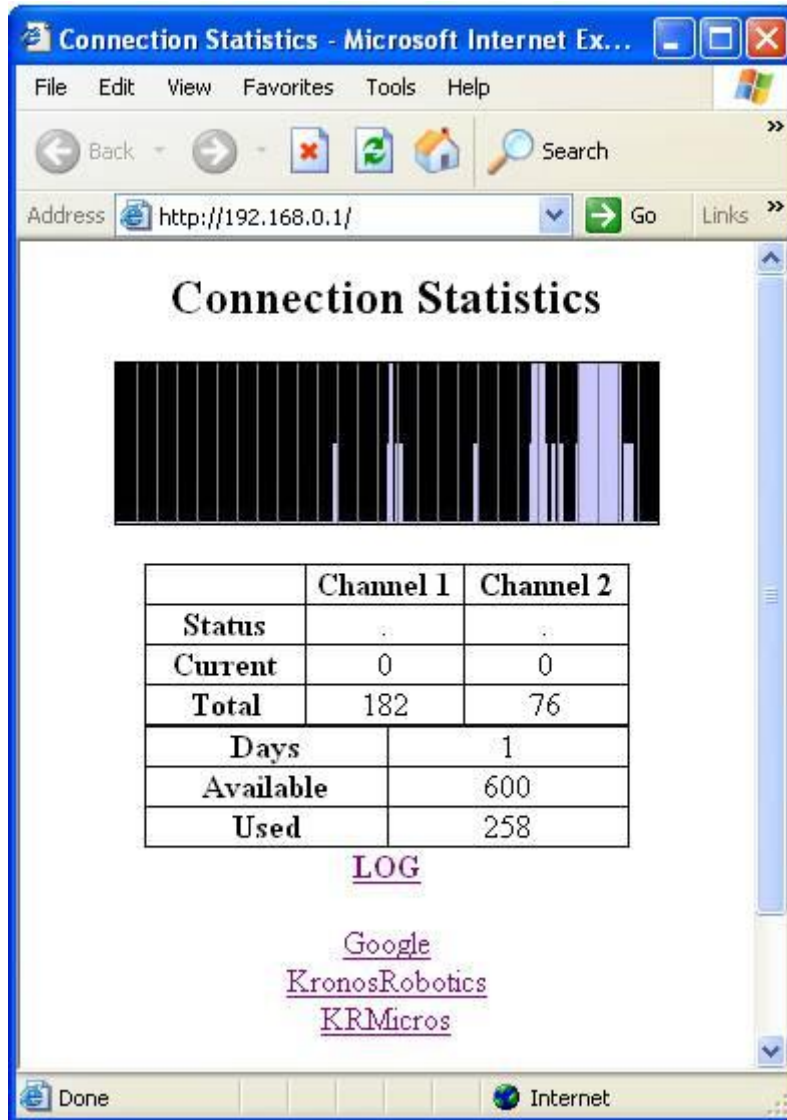


How to Build an Application Server

By Michael Simpson



One of the reasons we added sockets to Zeus was in order to create an application server. You might be asking yourself; just what is an application server?

An application server is a specialized program that performs an interface to an external device and serves it up on the Web. The device could be a washing machine or a industrial robot.

In the example that follows I will be using an ISDN router as an example. Many of you may be thinking just what is this ISDN thingy any way. ISDN was a system that allowed near broadband before DSL or Cable was around. And in many rural areas is still used. It doesn't really matter; all you need to know is that it has a RS232 interface for collecting data. In this example I am going to tap into this data and serve it up as a web page.

Step by Step

Step 1

The first thing you need to do in any application server design is to create the interface to the device that you want to control or monitor. You do this with the serial routines. Just create a nice little interface with all the controls and display values that you want to serve up. This will be about 90% of the task.

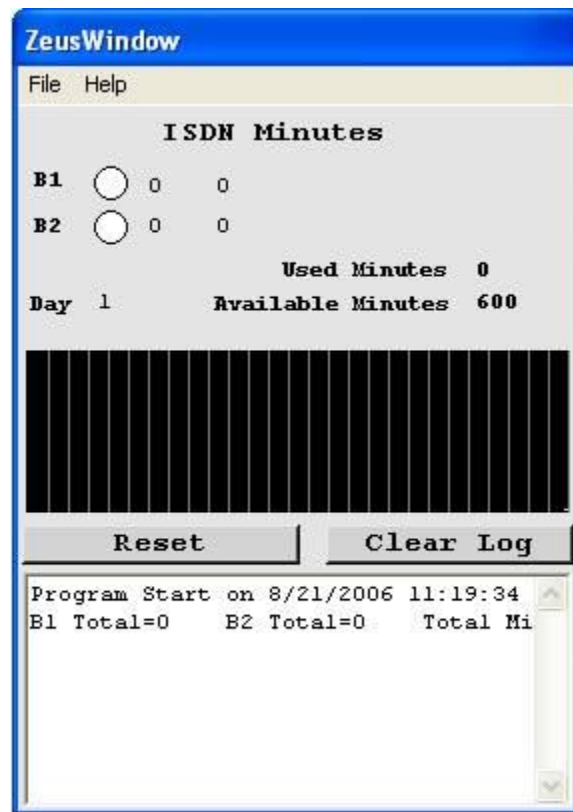


Figure 1

The program shown in Figure 1 simply connects to the device. In this case an ISDN router and collects the information and displays it on the form. An item that you may want to look at more specifically here is the bitmap generator as they could be used to display various amounts of scrolling data for other devices.

Before our main loop we create three bitmaps as shown in the code fragment shown here.

```

FormCreateBitMap(0,270,80)
FormCreateBitMap(1,270,80)
FormCreateBitMap(2,270,80)
FormUpdateAutoOff()
PlotBM(1)

```

The first two bitmaps are used to create a scrolling chart by the PlotBM function. What we do is plot out one bitmap and display it. The next time around we write that bitmap offsetting the horizontal axis to another bitmap. And add a new plot to the end and display it. The next time we reverse the two bitmaps and continue flipping as we go.

The third bitmap is used once we add our web server components so that we can create a picture to upload to the client.

The complete code can be found in the program file routmon1.txt.

Step 2

In the 2nd step we add the web server aspects of the code. The first piece is added just before our main loop as shown in this code fragment.

```

dim Sport as integer
Sport = 80 '----- Set Port here
print "Server Port = "+Sport

'-- Setup to listen for a connection
SocketListen(0,Sport,1,5)
Print "Waiting for Connections"

```

This bit of code creates the server listen port and starts the web server listening.

We then add the following bit of code to our main loop.

```

'-----
' Web Server
'-----
'-- Process Connections here --
if SocketError(0) > 0 then
    print "!!! Error "+SocketData(0,5)+" !!!"
    SocketClearError(0)
endif

if SocketQuickCheck(1) > -1 then
    For x = 1 to 5
        Sleep(2)
        if SocketState(x) = 4 and SocketBuff(x) > 0 then
            indata= SocketInput(x)
            'print x,indata

            'Lets break it down
            cmd = GetWord(indata,1,1," ")

```

```

file = GetWord(indata,1,2," ")
'Default
strif file = "" or file="/" then
    file = ".Router.htm"
endif
print x+" "+SocketData(x,11),cmd,file
strif cmd="GET" then
    tfile=ProcHTTPGet(file)
    SocketOutput(x, tfile)
    SocketClose(x)
endif
endif
Next
Endif

```

All this piece does is checks the 5 connections that we have allotted to this web server and passes any request to the handler routine called ProcHTTPGet. It also pulls the requested file name from the browser so that we can look at its name. If no specific page of file is requested we insert one. In this case we use the name “.Router.htm” as a default.

Note that if no requests come in from a web browser the program continues along as it did before.

The real processing is done by the ProcHTTPGet function shown here.

```

'-----
'-- HTTP Get
'-----
func ProcHTTPGet(filename as string) as string
    dim ctype as string
    dim FileData as string
    dim tmp as string
    dim x as integer
    FileData = GetFile(filename)
    strif FileData = "" then exit("HTTP/1.0 400 Error"+chr(13)+chr(10))

    if StrLookDown(-1,3,filename, ".htm") > -1 then ctype = "text/html"
    if StrLookDown(-1,3,filename, ".txt") > -1 then ctype = "text/plain"
    if StrLookDown(-1,3,filename, ".rtf") > -1 then ctype = "text/rtf"
    if StrLookDown(-1,3,filename, ".xml") > -1 then ctype = "text/xml"
    if StrLookDown(-1,3,filename, ".zip") > -1 then ctype = "application/zip"
    if StrLookDown(-1,3,filename, ".pdf") > -1 then ctype = "application/pdf"
    if StrLookDown(-1,3,filename, ".jpg") > -1 then ctype = "image/jpeg"
    if StrLookDown(-1,3,filename, ".bmp") > -1 then ctype = "image/bmp"
    if StrLookDown(-1,3,filename, ".gif") > -1 then ctype = "image/gif"
    if StrLookDown(-1,3,filename, ".gif") > -1 then ctype = "image/gif"
    if StrLookDown(-1,3,filename, ".mpg", ".mpeg", ".mp3") > -1 then
        ctype = "audio/mpeg"
    endif
    if StrLookDown(-1,3,filename, ".wav") > -1 then ctype = "audio/wav"

    '---Main File
    strif filename = ".Router.htm" then
        if blstat = 1 then
            FileData = Replace(FileData, "ZZZ1", "UP")
            FileData = Replace(FileData, "ZZZ3", bltcur)
        else
            FileData = Replace(FileData, "ZZZ1", ".")
            FileData = Replace(FileData, "ZZZ3", 0)
        end if
    end if
end func

```

```

endif
if b2stat = 1 then
  FileData = Replace(FileData,"ZZZ2","UP")
  FileData = Replace(FileData,"ZZZ4",b2tcur)
else
  FileData = Replace(FileData,"ZZZ2",".")
  FileData = Replace(FileData,"ZZZ4",0)
endif
FileData = Replace(FileData,"ZZZ5",b1dtot)
FileData = Replace(FileData,"ZZZ6",b2dtot)
FileData = Replace(FileData,"YYY3",int(b1dtot+b2dtot))
FileData = Replace(FileData,"YYY1",ddays)
FileData = Replace(FileData,"YYY2",davail)

if drawflag = 0 then
  BitMapDrawBitMap(1,2,0,0)
  FormPen(128,128,128)
  for x = 10 to 280 step 10
    BitMapLine(2,x,0,x,80)
  next
  FormBitMapSave(2,".Pic.jpg",3)
else
  BitMapDrawBitMap(0,2,0,0)
  FormPen(128,128,128)
  for x = 10 to 280 step 10
    BitMapLine(2,x,0,x,80)
  next
  FormBitMapSave(2,".Pic.jpg",3)
endif

endif

'---Log File
strif filename = "/log.htm" then
  tmp = FormTextBox(RM_Log)
  tmp = Replace(tmp,chr(13)+chr(10),"<br>"+chr(13)+chr(10))
  FileData = Replace(FileData,"XXX1",tmp)
endif

exit("HTTP/1.0 200 OK"+chr(13)+chr(10)+_
"Content-Type: "+ctype+chr(13)+chr(10)+"Content-Length: "+_
Len(FileData)+chr(13)+chr(10)+chr(13)+chr(10)+FileData)

endfunc

```

The real heart of this program is in the strif commands where we look for the .Router.htm and the /log.htm file names. We will do two different actions depending upon which request we get. You can create as many of these as you wish.

Before I get into what each of these actually do I need to explain the html section of the process. You will need to create your default html page. This can be done by hand if you know html or using FrontPage or some other tool.

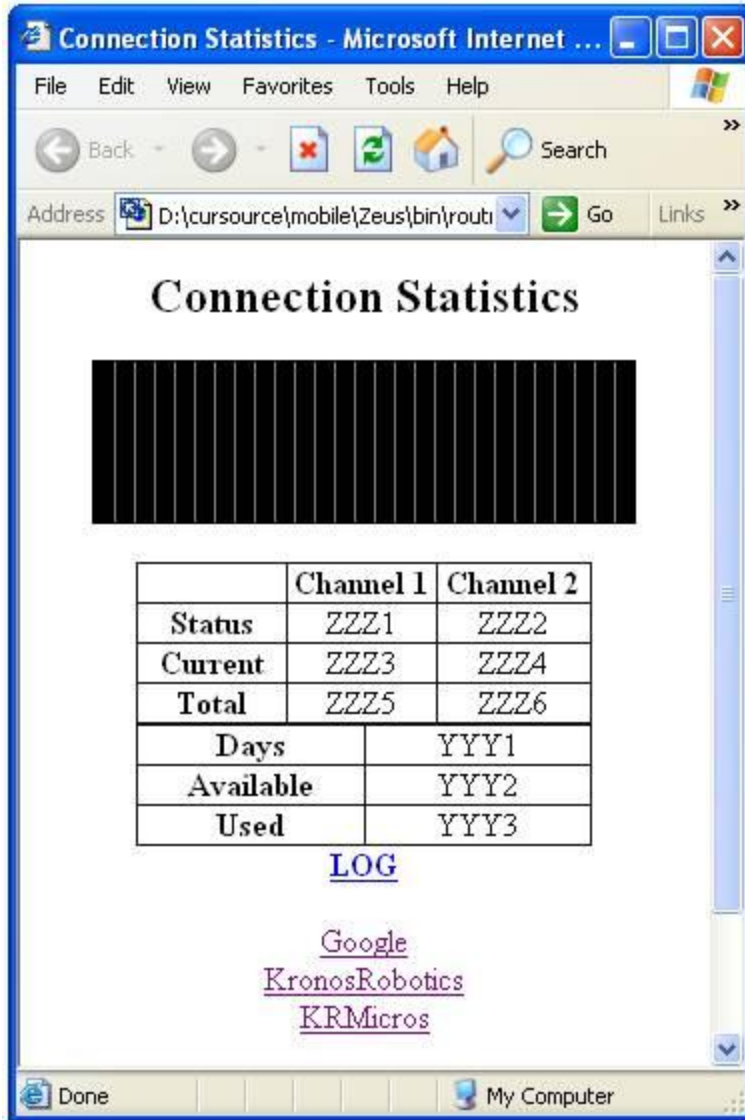


Figure 2

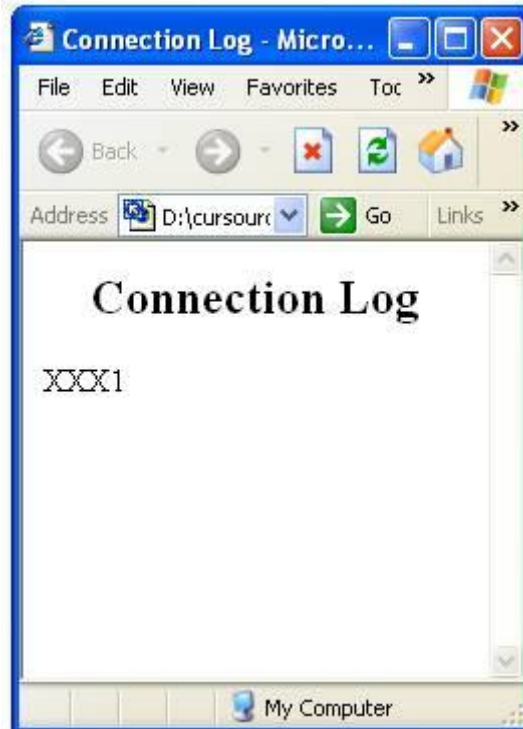


Figure 3

Notice in Figure 2 I created a Nice little page that has a couple of tables as well as a picture called Pic1.jpg. The tables contain some dummy data. Notice that each piece of data is different as in ZZZ1 and ZZZ2 and so on. This is so that we can replace that bit of data with our own when we serve up the page.

Figure 1 is the router page that I will serve up when **.Router.htm** is requested and Figure 3 is the page I will serve up when **/log.htm** is requested.

Now if you look back at the strif command where we check for **.Router.htm** you will see several Replace commands. With these commands we are replacing each piece of data with real-time data from our application. The same goes for the **/log.htm**.

One other important thing. We need to build the Pic1.jpg that is requested. This is done with the following code fragment:

```
if drawflag = 0 then
  BitMapDrawBitMap(1,2,0,0)
  FormPen(128,128,128)
  for x = 10 to 280 step 10
    BitMapLine(2,x,0,x,80)
  next
  FormBitMapSave(2,".Pic.jpg",3)
else
  BitMapDrawBitMap(0,2,0,0)
```

```
FormPen(128,128,128)
for x = 10 to 280 step 10
  BitMapLine(2,x,0,x,80)
next
FormBitmapSave(2,".Pic.jpg",3)
Endif
```

We draw to the bitmap and save it as a Jpeg file. This way when the browser requests the picture file it will get the latest one.

The completed server is included as routmonserver1.txt. I have also included the two htm files as well.

Sum it Up

That's how simple it is. In this example we give the user snapshots of real-time information. If you wanted to actually control buttons or change settings this could be done a couple different ways. A simple way would be to request a particular page and add a strif to trap it and issue what ever commands we wish. Another way is to add form buttons or options. You can see a example of this in the CMU cam server as well as the included code examples that came with Zeus.

To test the web server on your own machine use the localhost URL of <http://localhost>

Please note that if you already have a web server running on your machine you cant use port 80. Try using 8001 or something similar. If you do use a port number other than 80 you will have to include it in your URL as in <http://localhost:8001>