jBASE Web Builder – Integrating Visual Basic

INTRODUCTION

jBase for web builders is a fully featured environment for creating interactive and transactional web applications. Using the power of Microsoft's Visual Basic may extend this functionality.

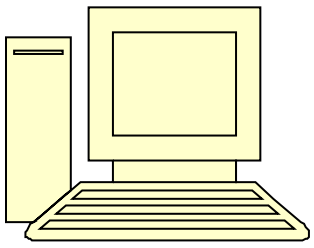
By integrating Microsoft Visual Basic in this way, the whole spectrum of ActiveX / COM components can be utilised in a jBASE for web builders implementation.

This document explains the principles behind this integration. It includes instructions on how to build a jWB project and a VB project from scratch which can interact together.

The example projects described in this document require jWB v3.2.1 or above, and Microsoft Visual Basic 6. These products need to be installed on the same machine. If they are on different machines, the IP address in the examples will need to be changed from 127.0.0.1 to the IP address of the VB server.

OVERVIEW

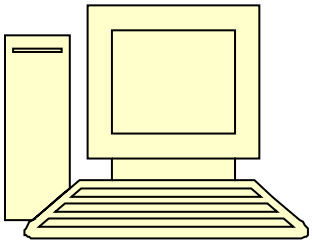
jBASE for web builders versions 3.2.1 and above come with a built in socket calling interface which allows communication via TCP/IP. By writing function calls using this socket interface, jWB can be easily integrated with other applications or programming languages.



JWB Server

1. jWB server sends a message via socket calling interface.

3. VB returns any results appropriate to jWB.



VB Server

2. VB server receives message, does any work required, including instantiating COM components.

Each conversation must be instantiated by jWB. Although the diagram shows the jWB server and the VB server as separate machines, they can co-exist on one machine.

BUILDING THE WEB BUILDER PROJECT

The web builder application is going to consist of a simple page consisting of a button and a label. The button will initiate the socket call to the Visual Basic server, and the label will display the message returned from the Visual Basic server.

Start up jWB and add a new Application Module. Give it a name of "socket". Add a new page to the module. Name the page "pagSocket".

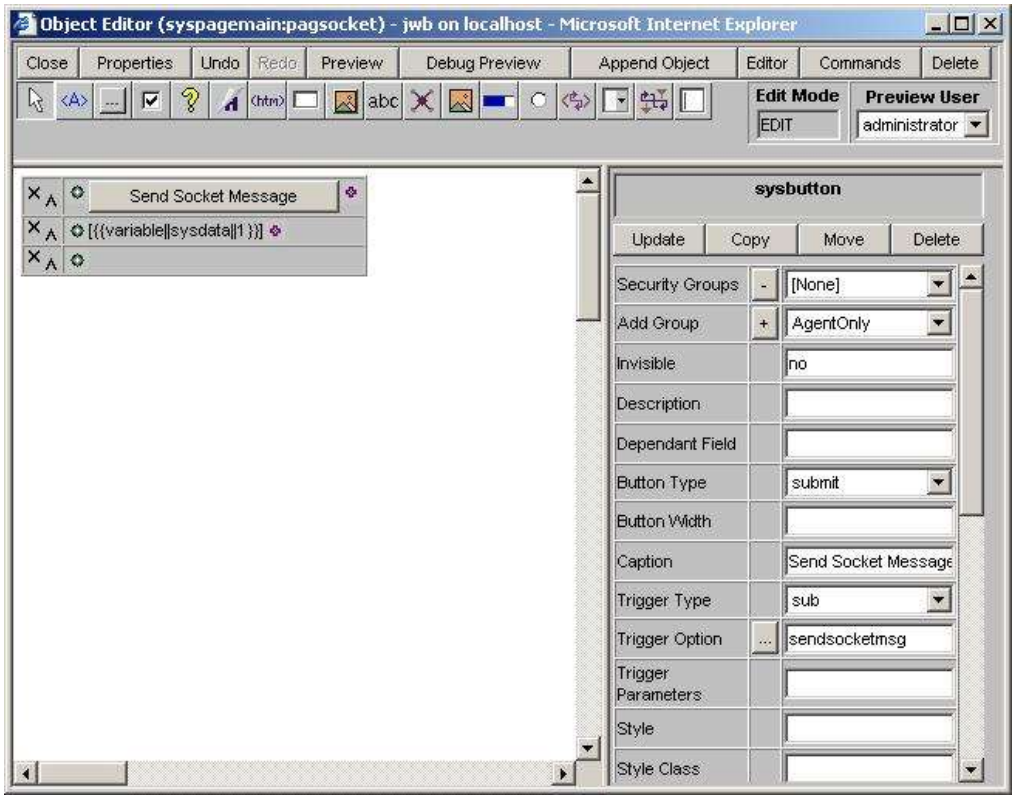
Add a button and a label to the page. Change the properties of the button as follows;

Caption	Send Socket Message
Button Type	Submit
Trigger Type	sub
Trigger Option	sendsocketmsg

Change the properties of the label as follows;

Text	[{{variable||sysdata||1}}]

The page should look like this;



Click on the Send Socket Message button, and then click on the ellipses button next to the trigger option parameter to open the sendsocketmsg routine.

Paste in the following routine;

```
     SUB sendsocketmsg(html)

*--- include the common block
     INCLUDE sysbp syscommon

*--- Build parameter list for syssocket call
     syssock<syssockhost> = "127.0.0.1"
     syssock<syssockport> = 8000
     syssock<syssockout> = "Message to VB"
     syssock<syssocktimeout> = 10

*--- Make socket call
     CALL syssocket

*--- Capture return value (or error)
     IF syserr <> 0 THEN
          sysdata<1> = syssock<syssockerrmsg>
     END ELSE
          sysdata<1> = syssock<syssockin>
     END

*--- Re-build jWB page
     CALL syscreate(syspage,html)

     RETURN
```

This routine builds a set of parameters before calling the socket function. When the socket function returns, it loads sysdata<1> with the value returned or an error message as appropriate.

Once sysdata<1> has been loaded, the page is re-created. The value of sysdata<1> will be displayed in the label on the page because of the DIRT tag.

The routine to make the socket call is syssocket. All parameters for the syssocket function are contained within the syssock common block.

The parameters for syssock are as follows;

syssock<syssockhost>	Hostname or IP address of machine running the server. For servers running on the same machine, use 127.0.0.1 or localhost.
syssock<syssockport>	TCP/IP port to connect to on the server. It is important that this port is not currently in use. Check http://www.iana.org/assignments/port-numbers for commonly used port numbers
syssock<syssocktimeout>	Timeout value in seconds for the conversation.
syssock<syssockout>	String value which will be sent to the server.
syssock<syssockin>	String value received from the server.
syssock<syssockerr>	If there has been an error interacting with the server, this will be set to non-zero.
syssock<syssockerrmsg>	If there has been an error interacting with the server, this will contain a description of the error.

BUILDING THE VISUAL BASIC SERVER

Open Visual Basic and create a new "Standard EXE" project. From the Project menu, choose the components option, and wait for the components dialog to appear. Scroll down, and tick the "Microsoft Winsock Control" component.

Rename form1 to frmSocket and change the caption property to Socket Form. Add a text box to the form. Change the multiline property of the text box to True and change the scrollbars property to "2 – Vertical". Delete the text property, and change the name to txtReceived.

Add two Winsock controls to the form. Change the names to WinSockGeneral and WinSockListener. On the WinSockGeneral control, change the LocalPort property to 8000.

The frmSocket form should look something like this;

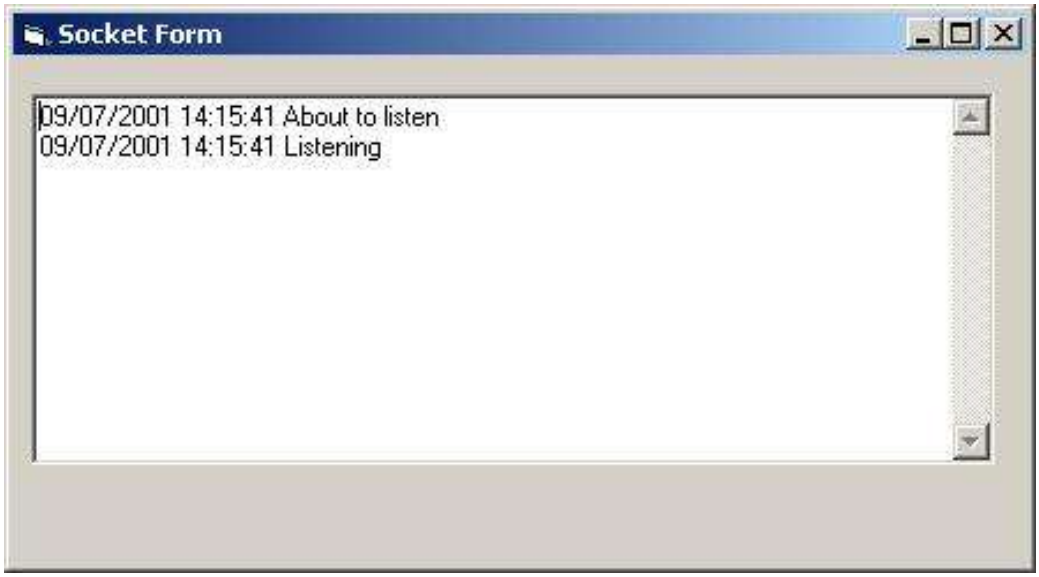
Add the following lines of code to the VB project; (Some lines may have scrolled onto more than one line in this document).

```
Option Explicit

Private Sub Form_Load()

    ' Start listening for socket messages
    txtReceived = txtReceived & Now & " About to listen" & vbCrLf
    WinsockGeneral.Listen
    txtReceived.Text = txtReceived.Text & Now & " Listening" & vbCrLf

End Sub

Private Sub WinSockGeneral_ConnectionRequest(ByVal requestID As Long)

    ' Answer the connection request
    txtReceived = txtReceived & Now & " Connection Request" & vbCrLf
    If WinSockListener.State <> sckClosed Then WinSockListener.Close
    WinSockListener.Accept requestid

End Sub

Private Sub winsocklistener_DataArrival(ByVal bytesTotal As Long)

    ' Get the message, and display it.
    Dim msg As String
    WinSockListener.GetData msg, vbString
    txtReceived = txtReceived & Now & " DataArrival : " & msg & vbCrLf

    ' Send a reply
    WinSockListener.SendData "Message back to jWB"

End Sub

Private Sub winsocklistener_Error(ByVal Number As Integer, Description
As String, ByVal Scode As Long, ByVal Source As String, ByVal HelpFile
As String, ByVal HelpContext As Long, CancelDisplay As Boolean)

    ' There's been an error so report it.
    txtReceived = txtReceived & Now & " Err: " & Description & vbCrLf

End Sub

Private Sub WinSockListener_SendComplete()

    ' Tidy up after message sent.
    txtReceived = txtReceived & Now & " Send Complete" & vbCrLf
    WinSockListener.Close

End Sub
```

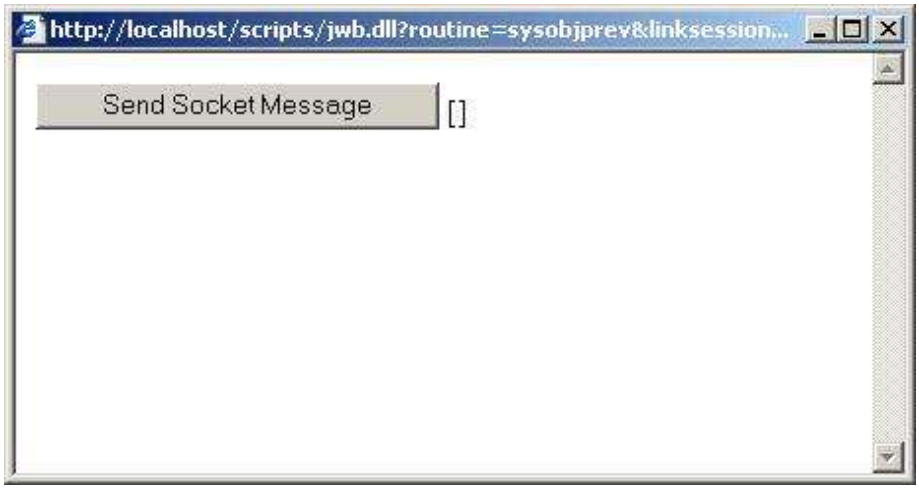
This code is designed to simply return a message to jWB. If the server was to carry out something more substantial, the WinSockListener_DataRetrieval routine would need to be extended to include whatever additional functionality was required.

TESTING THE TWO PROJECTS

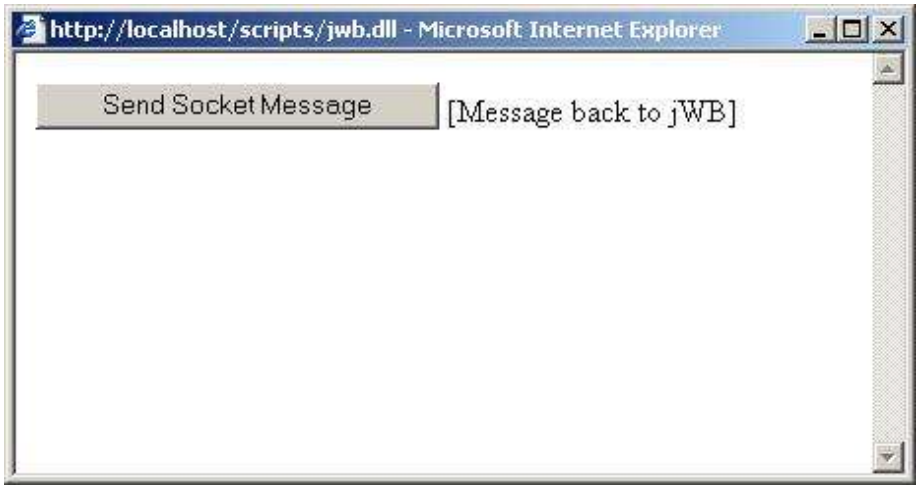
Start the Visual Basic project. After a brief pause, some text should appear in the text box on frmSocket. If everything is working correctly, the form should appear as follows;



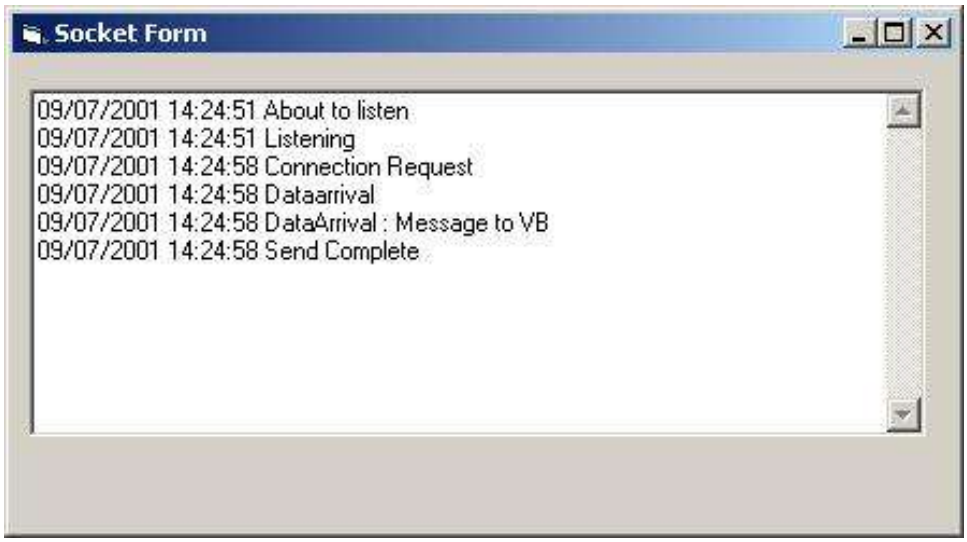
Open the web builder socket project, and load the pagsocket page. Click on the preview button. After a short pause, the page should appear as follows;



Click on the "Send Socket Message" button. After a short delay, a message should appear in the brackets adjacent to the button.



Switching to the Visual Basic program on the server should reveal more about the chain of events that caused the interaction between jWB and VB. As can be seen, there was a connection request event, followed by a Data arrival event and finally, there was a Send Complete event.



Document Created 9th July 2001
Author: Martin Bailey