GenBroker - OLE Automation Support

APPLICATIONS NOTE

August 2004

Description: GenBroker supports the reading and writing of OPC data through OLE automation! The functionality is available in both VBA and VBScript, and this applications note gives you a quick start on how to use both.

OS Requirement: Windows 2003/XP/2000/NT/95/98/Me.

General Requirement: General knowledge of scripting in VB and VBA, and an understanding of OPC, OLE automation, and GenBroker.

Using VBA

To use the GenBroker OLE automation features with VBA, you must first add a reference to your VBA project. In the VBA Editor, go to **Tools→References** and click the **browse** button to browse to **C: \Program Files\Common Files\ICONICS\GenClientWrapper.dll**.

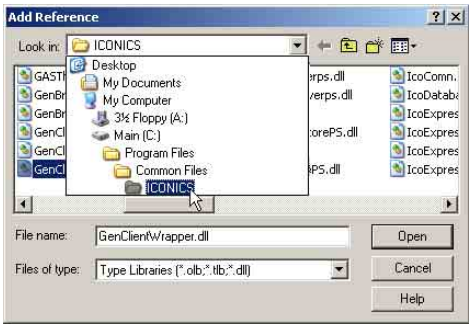

Fig. 1: GenClientWrapper.dll Location

Now the **GenClientWrapper** library and all of its functions are available to you!

The objects you have at your disposal are:

- **Client**
- **DataPoint**
- **SecurityPoint**

A **DataPoint** is what you would call an OPC tag, and each has the following properties and methods:

- **GetValueEtc (value, quality, timestamp, milliseconds)**
- **SyncWrite (newVal, [millisecondsTimeout=-1])**
- **Quality**
- **State**
- **Timestamp**
- **Value**
- **WritesPending**

To get you started, here is a simple script that reads a tag from the ICONICS OPC Simulator, then increments its value by 1:

```
Dim client As GENCLIENTWRAPPERLib.Client
Dim dp As GENCLIENTWRAPPERLib.DataPoint
Dim tag As String
Dim value As Variant, qual As Variant
Dim ts As Variant, tsms As Variant

Set client =
   CreateObject("GenClientWrapper.Client")
tag = "ICONICS.Simulator\SimulatePLC.OUTPUTS.
   FLOAT1"
Set dp = client.RequestDataPoint(tag, 100, 0)
While dp.State <> GC_POINT_OK_UPDATED
' Do nothing until point is ready
Wend
'Get tag information
dp.GetValueEtc value, qual, ts, tsms
MsgBox "Value: " & value & vbCrLf & "Quality:
   " & qual & vbCrLf & "Timestamp: " & ts &
   vbCrLf & "Milliseconds: " & tsms
dp.SyncWrite value + 1
```

VB Script

VB Script works a little differently than VBA. The following is the same example in VB Script:

```
Dim client
Dim dp
Dim tag
Dim value, qual, ts, tsms

Set client =
   CreateObject("GenClientWrapper.Client")
tag = "ICONICS.Simulator\SimulatePLC.OUTPUTS.
   FLOAT1"
Set dp = client.RequestDataPoint(tag,100,0)
While dp.State <> 3
' Do nothing until point is ready to be read
Wend
'Get tag information
dp.GetValueEtc value, qual, ts, tsms
MsgBox "Value: " & value & vbCrLf & "Quality:
   " & qual & vbCrLf & "Timestamp: " & ts &
   vbCrLf & "Milliseconds: " & tsms
dp.Value = value + 1
While dp.WritesPending
' Do nothing until value has been written
Wend
```

VBA/VB Script Differences

- **SyncWrite:** This works only in VBA. Writing in VB Script must be done directly through the **Value** property of each **DataPoint**.

  From the above example, in VBA the line of code used is:

  ```
  dp.SyncWrite value + 1
  ```

  But in VB Script the line of code is:

  ```
  dp.Value = value + 1
  ```

- **WritesPending:** This is intended for use in VB Scripts. Since the write update does not take place right away, there is an extra **WritesPending** property added so that you can delay the termination of your script until it is done writing the value. There is no need to do this in VBA when using **SyncWrite**.

  For example:

  ```
  While dp.WritesPending
   'Wait   for   value   to   be   written
  Wend
  ```

Performance Considerations

Another major difference to consider is that VBA supports global variables and VB Script does not.

This means that in VBA you can separate the script into three smaller scripts or procedures. For example the **GenClient** object is created and the **DataPoint**(s) are initialized in one event script such as **PreRuntimeStart**. Then the objects are manipulated in one or more runtime scripts (a timer ActiveX for example). Finally, the objects are destroyed in an event script such as **PostRuntimeStop**.

With VB Script, you must encapsulate all of this code in one script. While this works reliably, it is less efficient than the VBA method, and therefore you cannot expect the same performance out of your application.