Developing an OPC Client Application Using Visual Basic

Evan Cone and Heather Edwards

Introduction

This application note explains how to write programs that communicate with diverse devices, such as PLCs, RTUs, and FieldPoint devices, in programming environments such as Visual Basic and Visual C++. Historically, systems integrators have had to write proprietary or custom interfaces to retrieve data from devices manufactured by different hardware vendors. Now there is an industry-standard application programming interface (API) available so that you can create one client application that can communicate with different devices. This standard is known as OLE for process control (OPC). OPC is based on Microsoft’s Component Object Model (COM) and Object Linking and Embedding (OLE) technologies. If you are comfortable programming with COM, you can write programs using the OPC Automation API. Otherwise, you can take advantage of tools such as the Measurement Studio DataSocket ActiveX control, which provides an easy programmatic interface to OPC servers.

Note Measurement Studio includes tools to build measurement applications in ANSI C, Visual C++, and Visual Basic. The Measurement Studio Visual Basic tools were formerly known as ComponentWorks.

What Is OPC?

Before OPC, automation suppliers and systems integrators developed proprietary interfaces to their devices and control systems to permit access to real-time information. The problem with developing these proprietary interfaces is that users had to rewrite code for communicating with different devices because each one used a different API. Several years ago, Microsoft introduced OLE, COM, and distributed COM (DCOM) technologies; with them, application software can interoperate and communicate between modules distributed across a computer network. Automation hardware and software vendors, including National Instruments, formed an organization to define the standard for using COM in manufacturing applications. The result was OPC.

The OPC Foundation, which was chartered to develop the standard, now consists of more than 200 companies. The foundation’s goal is to develop a common interface for connectivity – an open and interoperable interface standard, based on the functional requirements of OLE and COM technologies. It will foster greater interoperability between automation and control applications, field systems and devices, and business and office applications.

For more information about OPC, visit ni.com/opc This Web site includes information about OPC and National Instruments OPC products. Follow the Measurement Studio DataSocket link to download OPC client examples written in Visual Basic and Visual C++.
Why Do We Need OPC Standards?

With hundreds of major hardware and software vendors, you have the difficult task of making sure that your client application can communicate with any device and driver. Before OPC, you had to write separate client application code to communicate with each device because each device driver used a different API. OPC provides an industry-standard interface – a single API – so client applications can retrieve process variables, such as temperature, pressure, flow rate, or position, and set control variables, such as the current or discrete output on an I/O module.

OPC offers the following benefits:

- Single API for all OPC servers so that you can reuse the code for your client application with each device.
- Opportunity to develop client applications in development environments that take advantage of COM and ActiveX, such as Microsoft Visual Basic, Visual C++, Excel, and Internet Explorer; and Borland Delphi. This application note explains how to implement an OPC client in Visual Basic.
- Browser through which you can select OPC items available to clients. An OPC item is a channel or variable in a real-world device (normally an I/O point) that a device server monitors or controls.
- Distributed and remote access through DCOM. You can access devices connected to other computers on the network.

How Can We Take Advantage of OPC?

If you are comfortable programming with COM, you can write programs using either the OPC Custom API or Automation API, depending on the programming environment in which you are developing your OPC client.

If you do not want to program with COM, use this application note to learn how to develop an OPC client example using the DataSocket control, an ActiveX control for sharing data between applications and devices. DataSocket offers the following benefits:

- Simple API, helpful to developers who don't want to program with COM.
- Access to the OPC server custom interface (a more efficient interface available from Visual C++), which enables your applications to run faster than applications that access the automation interface (the interface from Visual Basic or Borland Delphi). DataSocket is not an extra layer but rather a direct connection to OPC servers, and DataSocket always accesses the custom interface, regardless of the ActiveX container in which you are developing.
- Easy integration with National Instruments products such as LabVIEW.
What Is DataSocket?

DataSocket is a technology for sharing data between applications or different data sources. The DataSocket control provides a simple interface through which it can interact with OPC servers, such as National Instruments FieldPoint, from any ActiveX container, including Visual Basic and Visual C++. You can connect to an OPC server with DataSocket using an OPC URL, which is similar to the URLs used in a Web browser. URLs provide a standard mechanism for referring to locations. You already know how to use URLs to locate things on the Web, and you can locate OPC data items with DataSocket using a similar URL model:

```
opc://machine_name/server_name/item_name
```

<table>
<thead>
<tr>
<th>machine_name [optional]</th>
<th>Computer on which the OPC server is installed. DataSocket can access OPC servers on other computers using DCOM. If the machine name is omitted, DataSocket directly connects to the OPC server on the computer on which it is running.</th>
</tr>
</thead>
<tbody>
<tr>
<td>server_name</td>
<td>OPC server (provided with the hardware) to connect to.</td>
</tr>
<tr>
<td>item_name</td>
<td>OPC item on the specific OPC server. An item is a channel or variable (normally an I/O point) in a real-world device that a device server monitors or controls. Example FieldPoint item: FP Res\FP-DI-330 @1\Channel 1</td>
</tr>
<tr>
<td>UpdateRate=n [optional]</td>
<td>Maximum rate at which the OPC server will indicate that an item’s value has changed. The server should use an update rate as close as possible to the rate requested by the client. If this parameter is omitted, the default is 100 ms. The following example sets the update rate to 1000 ms: UpdateRate=1000</td>
</tr>
<tr>
<td>DeadBand=n [% of range] [optional]</td>
<td>Percentage change required before the server notifies your application of a value change. Not all servers support this option. If omitted, this parameter defaults to 0%. The following example sets the deadband to 10%: DeadBand=10</td>
</tr>
</tbody>
</table>

For example, the following URL connects to the sine item on the National Instruments OPCDemo server installed with Measurement Studio on the local machine. The options at the end of the URL specify an update rate of 1000 ms and a deadband of 10%.

```
opc://National Instruments.OPCDemo/sine?UpdateRate=1000&DeadBand=10
```

Example – Building a Client Application in Visual Basic

In this example, you will build a client application that reads from and writes to an OPC server. You also will display the quality and timestamp of the data and present a graphical browsing interface for users to select OPC items. Without changing any code, you can connect this client application to any OPC server.

This example is divided into the following parts:

- Open a new project and load the Measurement Studio ActiveX controls.
- Develop a client application that connects to an OPC server and retrieves and charts data from the OPC server. The client application will be able to connect to any installed OPC server.
- Display quality and timestamp.
- Browse for OPC items graphically.
- Write data to an OPC server.
- Connect to an OPC FieldPoint server or other OPC server (PLC or RTU) installed on your machine without writing any new code.

Note You can also download the Developing an OPC Client Application Using Visual Basic example from the Example Programs Database at ni.com/support
Opening a New Project and Loading ActiveX Controls

1. Launch Visual Basic from the Windows Start menu.
2. Open a new project. If you see the New Project dialog, open a Standard EXE. Otherwise, select File » New Project and select Standard EXE.
3. Right click on the Visual Basic Toolbox and select Components. If you don't see the Toolbox, select View » Toolbox.
4. Select National Instruments CW DataSocket and National Instruments CW UI. If the Measurement Studio controls are not in the Controls list, press the Browse button and select cwds.ocx and cwui.ocx from the Windows System directory.
5. Click OK.

Developing the OPC Client Application

To design the user interface, consider which tasks this application performs. The following list enumerates the tasks the client application needs to perform:

• Connect to the OPC server.
• Display the connection status.
• Retrieve and chart data from the OPC server.
• Disconnect from the OPC server.

Note To test the OPC client example, you need an OPC server. You can use the National Instruments OPC Demo server installed with Measurement Studio or you can use any other OPC server that you have installed on your computer. Some PLCs and RTUs come with their own OPC servers, or you can purchase OPC servers from third-party vendors such as National Instruments for many devices, including Allen-Bradley, Omron, and Optomux.
Figure 1 shows the OPC Client example.

1. Add the controls as shown in Figure 1.

   Note To place the controls on the form, select the icon from the Visual Basic Toolbox and click and drag to draw the control. After you place the control, you can use your mouse to move or resize the object. Right click on the object and select Properties to customize its appearance or behavior using the property pages.

   a. Place a CWGraph control on the form. Keep its default properties.

   b. Place a TextBox on the form. Right click on the TextBox and select Properties. Change the Name property to URLRead and the Text property to opc:/National Instruments.OPCDemo/sine. Add a Visual Basic Label to identify the TextBox control on the user interface. Change its Caption property to URL (Read).

   Note The URL points to a demo OPC server installed with Measurement Studio. Later in the example, you will learn how to browse for OPC servers so that you can connect to any installed on your computer.

   c. Place a Label on the form in which to display the status of the connection. In the Properties window, change the Name property to StatusRead and Border Style to Fixed Single. Delete the default text for the Caption property. Add another Visual Basic Label to identify this control on the user interface. Change its Caption property to Status (Read).

   d. Place a CommandButton control on the form. In the Properties window, change the Name property to ConnectBtn and the Caption to Connect to OPC Server.

   e. Place another CommandButton on the form and change Name to DisconnectBtn and Caption to Disconnect from OPC Server.
f. Place a CWDataSocket control on the form. You don’t need to set any properties for the DataSocket control.

2. Add Visual Basic code to make the application respond to interactions from users and the OPC server.

   Tip In Visual Basic, you can double click on a control to generate an event handler routine, which includes the control name, default event, and any parameters that are passed to the event handler routine. For example, to generate an event handler to connect to the OPC server, double click on the Connect to OPC Server button on the form.

   a. Users will connect to an OPC server by clicking on the Connect to OPC Server button. To generate an event handler, double click on the Connect to OPC Server button and add the following code:

      ```vba
      Private Sub ConnectBtn_Click()
        CWDataSocket1.ConnectTo URLRead.Text, cwdsReadAutoUpdate
      End Sub
      ```

      The ConnectTo method connects to the URL specified in the URLRead text box and automatically sends data to the client application as soon as it is available.

      Tip Use the Measurement Studio Online Reference to find information about the Measurement Studio ActiveX controls and their properties, methods, and events. You can launch the help from the Windows Start menu, the Measurement Studio custom property pages, or the Visual Basic Code window (highlight a property, method, or event and press <F1>).

   b. DataSocket automatically returns new data from the OPC server as soon as it is available. To chart the data as DataSocket returns it to the application, double click on the CWDataSocket control to generate the OnDataUpdated event and add the following code:

      ```vba
      Private Sub CWDataSocket1_OnDataUpdated(ByVal Data As CWDSLib.CWData)
        CWGraph1.ChartY CWDataSocket1.Data.Value
      End Sub
      ```

   c. To display the OPC server connection status (for example, whether DataSocket is connected or unconnected) in the StatusRead label, double click on the CWDataSocket control, select the OnStatusUpdated event from the top right list in the code window, and add the following code:

      ```vba
      Private Sub CWDataSocket1_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
        StatusRead.Caption = Message
      End Sub
      ```

   d. Users will press the Disconnect from OPC Server button to disconnect. To generate an event handler, double click on the Disconnect from OPC Server button and add the following code:

      ```vba
      Private Sub DisconnectBtn_Click()
        CWDataSocket1.Disconnect
      End Sub
      ```

3. Save the project and form as OPC Client.

4. Select Run»Start to run your program.

5. Press Connect to OPC Server to connect to the default OPC server, National Instruments.OPCDemo. DataSocket charts each data point as it is received. Notice that the status message confirms that DataSocket is currently connected to the OPC server.

6. Press Disconnect from OPC Server to disconnect DataSocket. Notice that the status message confirms that DataSocket is unconnected.

7. Select Run»End to close the OPC client application.
Checking the Quality and Timestamp of the Data

Because the OPC standard includes timestamp and quality information with data, you can display the quality and timestamp of the data. The Timestamp display shows the timestamp for each data point. The Quality LED remains green as long as the quality of the data is good. Figure 2 shows the OPC client with the Timestamp and Quality indicators.

![Figure 2. OPC Client with Timestamp and Quality](image)

1. Add the controls as shown in Figure 2.
   
   a. Place a Label control on the form to display the timestamp attribute. In the Properties window, set the **Name** property to **Timestamp** and the **Border Style** property to **Fixed Single**. Delete the default text in the **Caption** property.
   
   b. Add a Visual Basic label to identify the timestamp display. Change the **Caption** property to **Timestamp**.
   
   c. Place a CWButton control on the form. To configure a Measurement Studio control, whose name always begins with CW, you need to set properties in the Properties window in Visual Basic and the custom property pages. In the Properties window, change the **Name** property to **QualityLED**. To set the rest of the properties, right click on the control and select **Properties**.

<table>
<thead>
<tr>
<th>Property Page</th>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Style</td>
<td></td>
<td>RoundLED</td>
</tr>
<tr>
<td>Button</td>
<td>On color</td>
<td>green</td>
</tr>
<tr>
<td>Button</td>
<td>Off color</td>
<td>red</td>
</tr>
<tr>
<td>Advanced</td>
<td>State</td>
<td>On</td>
</tr>
</tbody>
</table>
d. Add another Visual Basic label to identify QualityLED. Change the **Caption** property to **Quality**.

2. To have DataSocket display the quality and timestamp attributes of the data it returns, add the bolded code to the CWDataSocket1_OnDataUpdated subroutine:

   ```vbscript
   Private Sub CWDataSocket1_OnDataUpdated(ByVal Data As CWDSLib.CWData)
   Dim Quality As CWDSOPCQuality
   CWGraph1.ChartY CWDataSocket1.Data.Value
   'Show quality and timestamp
   TimeStamp.Caption = Data.GetAttribute("TimeStamp", "Not Available").Value
   Quality = Data.GetAttribute("Quality", cwdsOPCBad).Value
   QualityLED.Value = (Quality = cwdsOPCGood)
   End Sub
   ```

3. Save the project.

4. Run the project to see how quality and timestamp work.

5. Stop the OPC client example.

**Browsing for OPC Items**

Rather than typing the URL for the OPC server, end users of this application can browse for the OPC item. Figure 3 shows the OPC client with browsing capabilities.

![Figure 3. OPC Client with Browsing Capabilities](image-url)
1. Add a CommandButton control to the form as shown in Figure 3, and set the Name property to BrowseBtn and the Caption property to Browse for OPC Item.

2. Double click on the Browse for OPC Item button and add the following code to the event handler:

   ```vbnet
   Private Sub BrowseBtn_Click()
       If (CWDataSocket1.SelectURL("opc:", "Select Read URL")) Then
           URLRead.Text = CWDataSocket1.URL
           'Connect to selected OPC item in read mode
           CWDataSocket1.ConnectTo URLRead.Text, cwdsReadAutoUpdate
       End If
   End Sub
   ```

3. Save the project.

4. Run the project to see how the browsing works. Select Browse for OPC Item.

5. Select your machine and National Instruments.OPCDemo, as shown in Figure 4.

   ![Figure 4. Browsing for the OPC Server](image)

6. Press OK.

7. Select the channel that you want to read, as shown in Figure 5.

   ![Figure 5. Selecting the Read Item](image)

8. Press OK. DataSocket connects to the OPC server, and the data is charted on the graph.

9. Stop the OPC client example.
Writing Data to OPC Servers

To effectively communicate with a device, you need to write to it as well as read from it. To write to the OPC server, you need another DataSocket control to connect in write mode, a status message to display the connection status, and a numeric edit box to specify the write value. You also should allow users to browse for the OPC item that they want to write to with the existing Browse for OPC Item button and disconnect from the OPC server with the existing Disconnect from OPC Server button. Figure 6 shows the OPC client with the write feature.

Figure 6. OPC Client Extended to Write to an OPC Item

1. Add the controls as shown in Figure 6.
   a. Place a TextBox on the form. Right click on the TextBox and select Properties. Change the Name property to URLWrite, and the Text property to opc:/National Instruments.OPCDemo/MotorRPMSetpoint. Add a label to identify this control. Change its Caption property to URL (Write).

   b. Place a Label on the form in which to display the status of the connection. In the Properties window, change the Name property to StatusWrite and Border Style to Fixed Single. Delete the default text for the Caption property. Add a label to identify this control. Change its Caption property to Status (Write).

   c. Place a CWNumEdit control on the form and change its name to WriteValue. Add a label to identify this control as Write Value.

   d. Place another CWDataSocket control on the form.
2. Add Visual Basic code to make the application respond to interactions from users and the OPC server.
   a. To connect both DataSocket controls when users press the Connect to OPC Server button, add the bolded code to the `ConnectBtn_Click` subroutine:

   ```vbnet
   Private Sub ConnectBtn_Click()
       CWDataSocket1.ConnectTo URLRead.Text, cwdsReadAutoUpdate
       CWDataSocket2.ConnectTo URLWrite.Text, cwdsWriteAutoUpdate
   End Sub
   ``

   b. To display the OPC server connection status, double click on CWDataSocket2 and add the following code to the `OnStatusUpdated` event:

   ```vbnet
   Private Sub CWDataSocket2_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal Message As String)
       StatusWrite.Caption = Message
   End Sub
   ```

   c. DataSocket should automatically write the value to the OPC server as soon as the user enters it. To write the value, double click on the CWNumEdit control and add the following code:

   ```vbnet
   Private Sub WriteValue_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As Boolean)
       CWDataSocket2.Data.Value = Value
   End Sub
   ```

   d. To chart the write value, you have to add a second plot to the graph through the property pages and one line of code to the program. Right click on the graph control and select Properties. On the Plots page, click the Add button to add a second plot and change the plot color to pink. Click OK.

   Add the bolded line of code to the `CWDataSocket1_OnDataUpdated` event:

   ```vbnet
   Private Sub CWDataSocket1_OnDataUpdated(ByVal Data As CWDSLib.CWData)
       Dim Quality As CWDSOPCQuality
       CWGraph1.ChartY CWDataSocket1.Data.Value
       CWGraph1.Plots(2).ChartY CWDataSocket2.Data.Value
       ' Show quality and timestamp
       TimeStamp.Caption = Data.GetAttribute("TimeStamp", "Not Available").Value
       Quality = Data.GetAttribute("Quality", cwdsOPCBad).Value
       QualityLED.Value = (Quality = cwdsOPCGood)
   End Sub
   ```

   e. Users will expect to browse for both URLs if they press the Browse for OPC Item button. To allow them to browse for both, add the bolded code to the `BrowseBtn_Click` subroutine:

   ```vbnet
   Private Sub BrowseBtn_Click()
       If (CWDataSocket1.SelectURL("opc:", "Select Read URL")) Then
           URLRead.Text = CWDataSocket1.URL
       ' Connect to selected OPC item in read mode
           CWDataSocket1.ConnectTo URLRead.Text, cwdsReadAutoUpdate
       End If

       If (CWDataSocket2.SelectURL("opc:", "Select Write URL")) Then
           URLWrite.Text = CWDataSocket2.URL
       ' Connect to selected OPC item in write mode
           CWDataSocket2.ConnectTo URLWrite.Text, cwdsWriteAutoUpdate
       End If
   End Sub
   ```

   f. When users press the Disconnect from OPC Server button, they will expect both servers to be disconnected. To disconnect the second OPC server, add the bolded code to the `DisconnectBtn_Click` subroutine:
Private Sub DisconnectBtn_Click()
    CWDataSocket1.Disconnect
    CWDataSocket2.Disconnect
End Sub

3. Select Run » Start to run the OPC client.
4. Change the URL (Read) to opc:/National Instruments.OPCDemo/MotorRPMSensor.
5. Press the Connect to OPC Server button.
6. Try writing different values to the OPC server. Notice how the chart data adjusts to approach the setpoint you specify. When the value reaches the setpoint, no more data is charted because no more events are being fired, but you can continue to adjust the write value and see the adjustment reflected in the chart.

Figure 7 shows the finished OPC client application.

![Figure 7. Finished OPC Client Application](image)

7. Save the project.
Connecting to an OPC FieldPoint Server

You can connect to any OPC server using an OPC URL without having to change one line of code. For this example, use the browse feature to construct the URL of the OPC server that you want to connect to.

**Note** Before you can connect to the FieldPoint server with this client application, you need to create a configuration file (.iak) in the FieldPoint Explorer. For help with creating a configuration file, refer to the FieldPoint online help. You can download the latest version of the FieldPoint server from the National Instruments FieldPoint Web site at [ni.com/fieldpoint](http://ni.com/fieldpoint).

1. Press the **Browse for OPC Item** button. The Browse feature is going to let you browse for both URLs (the Read and Write URL).
2. Select your machine.
3. Select National Instruments.OPCFieldPoint. Figure 8 shows the OPC server browser for the read item with a selected machine and OPC server.

**Tip** You installed a demo OPC server when you installed DataSocket. You can connect to National Instruments.OPCDemo if you don’t have another OPC server installed on your computer.

4. Press **OK**.
5. Select the channel that you want to read.
6. Press **OK** to select the read item and connect to it.
7. Repeat steps 2–6 to browse for the OPC item that you want to write to and connect to it.

**Conclusion**

The OPC standard provides a common interface for connecting control applications to field systems and devices. DataSocket makes OPC easy to use by offering a simple API. Because DataSocket is an ActiveX control, you can use it in any ActiveX container, such as Visual C++, Excel, and Borland Delphi, to quickly and easily develop acquisition and control applications. For example, you can use the DataSocket ActiveX control in an Excel spreadsheet to import data from a device directly into Excel for analysis. Or you can create a component with DataSocket that acquires and displays live data on a Web page. Refer to the Application Note 127, *Building an Interactive Web Page with DataSocket*, for information on creating a component that you can embed in an HTML document. To access application notes, visit the National Instruments Web site at [ni.com](http://ni.com).

For more information about OPC, visit the National Instruments OPC Web site at [ni.com/opc](http://ni.com/opc).