Introduction

DISCLAIMER: This software is provided "as is" and is not supported by the Product Support Group at Opto 22. It is intended only as a starting point for writing your own client application.

The Easy OPC VB Forms provide class objects and collection objects to make it easy to connect to Opto 22's OptoOPCServer using Visual Basic.

There are two objects that expose all the events and methods you will need in order to connect to a server, add groups and items, and receive data from those items. Depending on your needs, you will use only one of the objects.

- Use the **ServerCollection** class object to gather data from multiple servers.
- Use the **CServer** class object to gather data from a single server.

It is up to you to construct a user-interface that meets the requirements of your application, but three important dialogs are provided to save you time:

- The **Add Server** dialog allows you to choose a server and the machine it resides on.
- The **Add Group** dialog allows you to add one or more groups to the server.
- The **Add Items** dialog allows you to add items to a group.

To help get you started, the examples below show how to use the events and methods of the ServerCollection class and the CServer class. A complete list of the events and methods is included for each object.

Before You Begin

In order for the VB6 OPC Client (TestOpcClient.vbp) to have data to work with, you *must use* Opto Browser Configurator to set up a list of available items. For information on how to do this, see form 1439, the *OptoOPCServer User's Guide*.

Using Group Handles and Item Handles

The underlying system generates the server handles, group handles, and item handles for you. Handles are simply unique numeric identifiers for your objects. They are also structured to make things as easy as possible for you. Note that the handle management system imposes a maximum limit on your overall system of 15 servers, 4095 groups per server, and 65535 items per group.

Each group handle contains the handle value of the server object that the group belongs to. Likewise, each item handle contains the handle value of the group object that the item belongs to. Therefore, it also contains the handle value of the server object that the item belongs to.

How this information is stored is unimportant and subject to change. What you do need to know is that there are some utility functions to help you work with and take advantage of the structure of handles. GetServerHandleFromGroupHandle( ) takes a

group handle as its input and returns its server handle. This enables you to use a group handle to determine the handle of its server object.

Two other similar functions, GetServerHandleFromItemHandle( ) and GetGroupHandleFromItemHandle( ) enable you to get the type of handle you need from whatever type of handle that you have. Used in conjunction with three other utility functions, GetServer( ), GetGroup( ), and GetItem( ), all of which return objects of their respective type, you have an easy way to work with objects with minimal effort and code.

Using the ServerCollection Class Object

Use the ServerCollection object to connect to one or more servers. This is the recommended way to get started.

1. In the main form of your application, add a variable of type ServerCollection. Make sure it is declared WithEvents.

```
Public WithEvents MyServers As ServerCollection
```

2. Implement a user-interface component like a menu item or a button that will enable you to call the Add Server dialog using the function DialogAddServer.

   The two parameters of DialogAddServer will be filled in with the server's name (ProgID) and the machine name that the server resides on (if any). The function returns a server handle. It is important that you store this server handle as it will be needed for input into other functions.

   In the example below, listviewServers is a ListView control that has been placed on the main form.

```
'
' cmdAddServer_Click
'
Private Sub cmdAddServer_Click()
  Dim nServerHandle As Long
  Dim sServerHandle As String
  Dim TheListItem As ListItem
  Dim sServerName As String
  Dim sNetworkNode As String

  ' Display the Add Server Dialog
  nServerHandle = MyServers.DialogAddServer(sServerName, sNetworkNode)
  If nServerHandle > 0 Then
    ' Add the server to the Servers listview
    sServerHandle = CStr(nServerHandle)
    Set TheListItem = listviewServers.ListItems.Add(, , sServerName)
    TheListItem.Tag = nServerHandle
    TheListItem.EnsureVisible
    TheListItem.Selected = True
  End If

End Sub
```

3. Now that you are connected to a server, you need to add one or more groups to the server.

   Implement user-interface components that will enable you to call the Add Group dialog using the function DialogAddGroup. The first parameter of the DialogAddGroup function is the server handle from Step 2. The other parameters of the DialogAddGroup function will be filled in with information supplied interactively by the user. You can use this information in any way you want. You do not have to save it as you will be able to look it up later if you need it as long as you save the group handle which is the return value of the function.

   In the example below, listviewGroups is a ListView control that has been placed on the main form and is used to store the name of the group and its associated group handle.

```
'
' cmdAddGroup_Click
'
Private Sub cmdAddGroup_Click()
  Dim TheListItem As ListItem
  Dim nGroupHandle As Long

  ' Make sure a server is selected
  Set TheListItem = listviewServers.SelectedItem
  If Not TheListItem Is Nothing Then
    Dim nServerHandle As Long
    Dim sGroupName As String
    Dim nUpdateRate As Long
    Dim nTimeBias As Long
    Dim rDeadband As Single
    Dim bIsActive As Boolean
    Dim nDataChangeMethod As Integer

    nServerHandle = TheListItem.Tag
 nGroupHandle = MyServers.DialogAddGroup(nServerHandle, sGroupName,_
                                                        nUpdateRate,
                                                        nTimeBias,_
                                                        rDeadband,
                                                        bIsActive,_
                                                        nDataChangeMeth
                                                        od)
    If nGroupHandle > 0 Then
      ' Add the group to the Groups listview
      Set TheListItem = listviewGroups.ListItems.Add(, , sGroupName)
      TheListItem.Tag = nGroupHandle
      TheListItem.EnsureVisible
      TheListItem.Selected = True
    Else
      If nGroupHandle = OPCDuplicateName Then
        MsgBox ("There is already a group by that name.")
      Else
        MsgBox ("There was an error while adding the group: " & _
                    Hex(nGroupHandle))
      End If
    End If
  End If

End Sub
```

4. Now that you have one or more groups, you need to add items to a group.

   Implement user-interface components that will enable you to call the Add Items dialog using the function DialogAddItems. The first parameter of the DialogAddItems

function is the group handle from Step 3. The return value from DialogAddItems is True if the user pressed OK in the dialog box and False otherwise. The output parameters in the DialogAddItems function call are filled in with information for each item chosen to be added by the user.

Since there can be more than one item added at a time, you must declare empty arrays for the item handles, item IDs, access paths, datatypes, and active states. The number of items is returned in an output parameter as well. There is also an error array that tells whether each item was added successfully or unsuccessfully.

This is all illustrated in the sample code below. Remember that it is essential to save the item handles for use as inputs to other functions.

```
'
' cmdAddItems_Click
'
Private Sub cmdAddItems_Click()
  Dim TheListItem As ListItem

  ' Make sure a group is selected
  Set TheListItem = listviewGroups.SelectedItem
  If Not TheListItem Is Nothing Then
    Dim nGroupHandle As Long
    Dim bRetval As Boolean
    Dim nNumItems As Long
    Dim sItemIDs() As String
    Dim bActiveStates() As Boolean
    Dim nItemHandles() As Long
    Dim nErrors() As Long
    Dim nDatatypes() As Integer
    Dim sAccessPaths() As String

    nGroupHandle = TheListItem.Tag
    bRetval = MyServers.DialogAddItems(nGroupHandle, nNumItems,
sItemIDs,_ bActiveStates, nDatatypes,_ sAccessPaths, nItemHandles,_
nErrors)
    ' DialogAddItems returns True if all items were added successfully,
    ' but it returns False if even one item had an error. So regardless
of
    ' the return value, it is a good idea to spin through the nErrors
    ' array.
    Dim sServerName As String
    Dim sGroupName As String
    Dim i As Integer

    sServerName = listviewServers.SelectedItem.Text
    sGroupName = listviewGroups.SelectedItem.Text

       ' Put the items that were successfully added into the Items
listview
    For i = 1 To nNumItems
      If nErrors(i) = 0 Then
        Set TheListItem = listviewItems.ListItems.Add(, , sServerName)
        TheListItem.SubItems(1) = sGroupName
        TheListItem.SubItems(2) = sItemIDs(i)
        TheListItem.Tag = nItemHandles(i)
      End If
    Next
  End If

End Sub
```

5. Once active items have been added, you will begin to receive data based on the update rate of the groups.

Form 1536-110804 • Opto 22 • 43044 Business Park Drive • Temecula, CA 92590-3614 • **www.opto22.com**
SALES 800-321-6786 • 951-695-3000 • FAX 951-695-3095 • sales@opto22.com • **SUPPORT** 800-835-6786 • 951-695-3080 • FAX 951-695-3017 • support@opto22.com

In order to catch the data and do something with it, you must implement the GlobalDataChange event that is associated with the ServerCollection variable that was declared in Step 1. Data for all groups and servers will be received in this one function. However, in any one GlobalDataChange event, information about the items in one group of one server is supplied. In the example below, the value, quality and timestamp of each item is displayed in the third, fourth, and fifth subitems of a ListView control named listviewItems that has been placed on the main form.

```
'
' GlobalDataChange
'
Private Sub MyServers_GlobalDataChange(ByVal nTransactionID As Long, _
                                       ByVal nServerHandle As Long, _
                                       ByVal nGroupHandle As Long, _
                                       ByVal nNumItems As Long, _
                                       nItemHandles() As Long, _
                                       vItemValues() As Variant, _
                                       nQualities() As Long, _
                                       TimeStamps() As Date)
  Dim i As Integer
  Dim j As Integer
  Dim nNumListviewItems As Integer
  Dim TheListItem As ListItem
  Dim nListItemHandle As Long

  nNumListviewItems = listviewItems.ListItems.Count

  For i = 1 To nNumItems
    For j = 1 To nNumListviewItems
      Set TheListItem = listviewItems.ListItems(j)
      If TheListItem.Tag = nItemHandles(i) Then
        ' Update the Value (3), Quality (4) and TimeStamp (5)
        TheListItem.SubItems(3) = CStr(vItemValues(i))
        TheListItem.SubItems(4) = ConvertQualityToString(nQualities(i))
        TheListItem.SubItems(5) = CStr(TimeStamps(i))
        Exit For
      End If

      Set TheListItem = Nothing
    Next
  Next

End Sub
```

How much functionality you implement depends on your application's needs. Refer to the description of each method or event in the ServerCollection source code for more information.

Events and Methods Exposed by the ServerCollection Object

Listed below are the events and methods that are exposed by the ServerCollection object.

- **Events**: GlobalDataChange, ServerShuttingDown, AsyncWriteComplete, AsyncCancelComplete
- **Methods**: DialogAddServer, AddServer, RemoveServer, GetServer, DialogAddGroup, AddGroup, DialogModifyGroup, RemoveGroup, GetGroup, DialogAddItems, AddItems, RemoveItem, GetItem, ReadItem, ReadItems, WriteItem, WriteItems, AsyncReadItems, AsyncWriteItems, AsyncRefresh, AsyncCancel

Using the CServer Class Object

This class can be used if you only want to connect to and gather data from one server.

1. In the main form of your application, add a variable of type CServer. Make sure it is declared WithEvents.

```
Public WithEvents MyServer As CServer
```

2. The next step is to connect to the server using the CServer class method ConnectToServer( ).

   Before calling this function, you will need to create a CServer object in one of two ways. If you know the ProgID and machine name where the server resides, the first code snippet below will fill your needs. If on the other hand you would like to display the Add Server dialog to allow your user to select a server, the second code snippet is what you will need. The global function DialogCreateServer( ) will display the Add Server dialog and return a CServer object.

```
' Create a new CServer object
Set MyServer = New CServer
If Not MyServer Is Nothing Then
  m_bConnected = MyServer.ConnectToServer("Opto22.OpcServer.2", "")
  If m_bConnected = True Then
    …
  Endif
Endif

' Display the Add Server dialog
Dim sServerName As String
Dim sNetworkNode As String

Set MyServer = DialogCreateServer(sServerName, sNetworkNode)
If Not MyServer Is Nothing Then
  m_bConnected = MyServer.ConnectToServer(sServerName, sNetworkNode)
  If m_bConnected = True Then
    …
  Endif
Endif
```

3. Once you are connected to a server, you will need to add groups and items. Follow the instructions in step 3 through step 5 above with one exception. In step 3, you will be calling the function DialogAddGroup in the CServer class. This function does not have a server handle as its first parameter. All other parameters are identical. The example source code below shows this.

```
'
' cmdAddGroup_Click
'
Private Sub cmdAddGroup_Click()
  Dim TheListItem As ListItem
  Dim nGroupHandle As Long

  ' Make sure we are connected to a server
  If m_bConnected = True Then
    Dim sGroupName As String
    Dim nUpdateRate As Long
    Dim nTimeBias As Long
    Dim rDeadband As Single
    Dim bIsActive As Boolean
    Dim nDataChangeMethod As Integer

    nGroupHandle = MyServers.DialogAddGroup(sGroupName, nUpdateRate, _
nTimeBias, rDeadband, _ bIsActive, nDataChangeMethod)
    If nGroupHandle > 0 Then
      ' Add the group to the Groups listview
      Set TheListItem = listviewGroups.ListItems.Add(, , sGroupName)
      TheListItem.Tag = nGroupHandle
      TheListItem.EnsureVisible
      TheListItem.Selected = True
    Else
      If nGroupHandle = OPCDuplicateName Then
        MsgBox ("There is already a group by that name.")
      Else
          MsgBox ("There was an error while adding the
                   group: " & _ Hex(nGroupHandle))
      End If
    End If
  End If

End Sub
```

Events and Methods Exposed by the CServer Object

Listed below are the events and methods that are exposed by the CServer object.

- **Events**: GlobalDataChange, ServerShuttingDown, AsyncWriteComplete, AsyncCancelComplete
- **Methods**: ConnectToServer, DisconnectFromServer, DialogAddGroup, AddGroup, DialogModifyGroup, RemoveGroup, GetGroup, DialogAddItems, AddItems, RemoveItem, GetItem, ReadItem, ReadItems, WriteItem, WriteItems, AsyncReadItems, AsyncWriteItems, AsyncRefresh, AsyncCancel