



# Visual Basic and OPC

All versions of Kepware OPC Servers

10/23/03 – v. 1.02

## Introduction

This document is intended to show the user the minimum required steps to connect an Visual Basic application to and OPC server and collect data. It is not intended to teach beginner or novice programmers how to create OPC based Visual Basic applications or how to program in VB. The programming styles used in the examples are those preferred by the document writer and may not follow standard programming practices.

## Recommendations

It is recommended that you be at least in intermediate level VB Programmer with a good understanding of Object Oriented process when doing this type of VB programming. You should print out and read the OPC Automation Interface Specification manual that is provided with the KEPServerEX installation. Lastly we do not currently support VB .Net so you should plan on using VB 6 or VB 5.

## About OPC

Unlike DDE where you would create a connection to the server for each control you created in your project OPC, which is OLE for Process Control, is a group of related interfaces that all the client application broader interface capacity then DDE does. The interface from VB is comprised of 4 objects and 2 object collections.

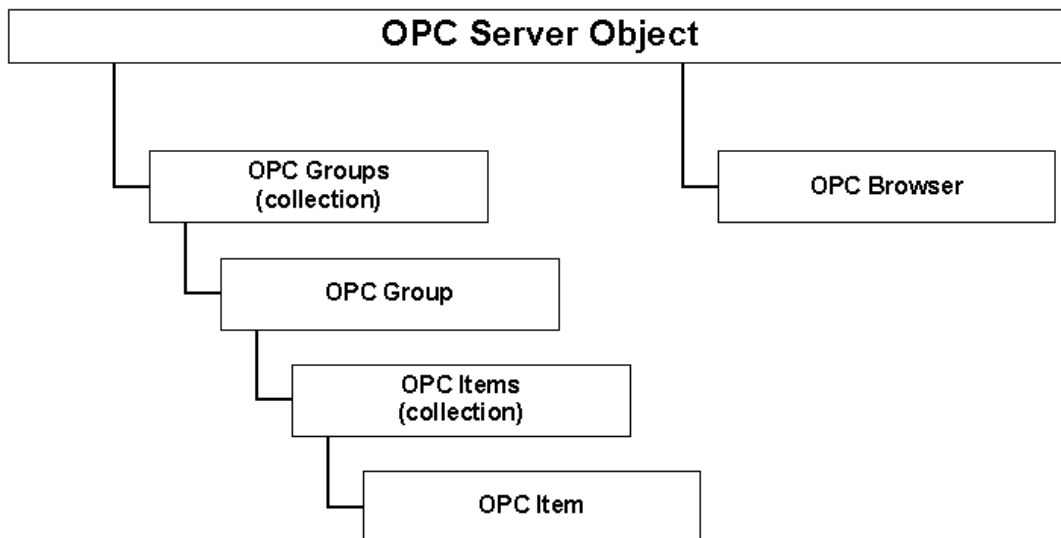


Figure 1

You can learn more about the structure of OPC and how it works from the KEPServerEX help file and the OPC Foundation web page at [www.opcfoundation.org](http://www.opcfoundation.org).

Now, using the absolute minimum number of code calls we will show you how to connect to and get data from and OPC server and display it in real-time on the form. The remainder of this document is dedicated to that process.

## **Creating the VB Application**

### ***Install KEPServerEX***

The first thing you will need to do if you have not already done so is install KEPServerEX on your PC. We recommend that you start with it on the same PC that you will be creating the VB app on. You do not need a license, the demo mode is full functioning and only needs to be restarted every 2 hours. In our example we are using the Simulator driver and the default server project of SimDemo.opf which installs with the server.

### ***Start Visual Basic***

Next you will start your Visual Basic Software. Create a new VB Executable project. You can name it whatever you wish. You will automatically get a new form.

### ***Adding Objects to the Form***

For this example we need to add two objects to the form.

The first object will be a text box. Name the text box "MyText" in the Name property. We will use the text box to display the value of item we are reading from the server.

The second object you will add to the form is a command button. Name the button "ExitExample" in the Name property. Make the caption for the button be "Exit" by typing "E&xit" in the Caption property. The button will be used to show you how to properly close a server connection.

### ***Referencing the OPC Automation Wrapper***

There is no way for VB to directly access the server via OPC so you will need to include a DLL in your project that will provide the software hooks you need. As part of the foundation we provide the generic OPC Automation Wrapper which is recompiled to ensure that it will connect to your server. The interface is a standard of the OPC foundation so in theory you should be able to use any manufacturer's interface to connect to any OPC Server. Our version of the file is automatically installed with the server and is located in the system32 folder of the Operating System root directory. The file is named "KEPOPCDAAUTO.dll".

In your project click on Project|References... in the VB Main menu to open the References dialog box and check the Kepware OPC Automation 2.0 Check Box.

### ***Adding the Code***

Now you need to start adding the code to the project so that you can read data from the server. In this example you will use a Server Object, a Group Collection Object, a Group Object, and an Item Collection Object.

#### **Project Declarations Code**

Ok, first we need to declare the global objects and variables that we are going to use in this project. This is pretty basic stuff, first you are going to force the declaration of all variables with the use of the "Option Explicit" statement. Next you are you will set the lower bound of any form Array's to 1 with the "Option Base" statement.

The rest of the declarations are all directly related to the OPC connection. You will be creating a Server Object, a Group Collection Object, a Group Object, and an Item Object Collection. You will notice that two of them are created using the " WithEvents " option. This is so the server can call back to the client project and inform it of changes in the data. This is often referred to as a callback event or exception event. This method is used so that the server does not have to send data to the client application that has not changed.. This allows more clients to be connected to the server application with less impact on system and network resources.

Lastly you will see that we have created several variables that are used as parameters of the various objects.

In your project open the form code window and enter the code located in Figure 1 into the General Declarations portion of the code window.

```
Option Explicit
Option Base 1

Dim WithEvents ConnectedOPCServer As OPCServer

Dim ConnectedServerGroups As OPCGroups
Dim WithEvents ConnectedGroup As OPCGroup
' OPC Item related data
Dim OPCItemCollection As OPCItems
Dim ItemCount As Long
Dim OPCItemIDs(1) As String
Dim ItemServerHandles() As Long
Dim ItemServerErrors() As Long
Dim ClientHandles(1) As Long
```

Figure 2

## Form Load Code

Now that we have defined the Objects it is time to start building the project. As we stated at the beginning this the absolute simplest example of connecting to an OPC Server and displaying data on the screen. To do that we are executing all the code needed to connect to the OPC Server and establish the data polling in the Form Load Sub Routine of the Form.

Create the Form Load Sub by either typing in the Form Code Window or selecting them from the drop downs at the top of the window.

**Note: A quick note about the naming conventions that we are using for the Objects and variables. These are basic standards that have been adopted here but you can use any names you wish. The names that are used are formatted to give an explanation of what they are used for.**

Ok First we create the server using the Set command. You will notice that we named ours "ConnectedOPCServer". You will then use the Server objects Connect method to connect to the server. Again you will notice that we are passing the ProgramID for KEPServerEX which is "Kepware.KEPServerEX.V4".

**Note: If you read the OPC Automation Interface Manual you will notice that we are only using one of the connection parameters. The "NodeName" parameter is optional and there is not needed when connecting to a Server running on the same PC as the VB project.**

Next you will create a Group collection to the Server object and then add a named Group to the Group collection. We named our Group "Test1". Now you will set the Group Update and subscription properties. We chose a fairly slow update rate of "500 msec" and set the subscription property to "True".

**Note: the group is the key to getting data. In OPC the client dictates the rate at which the server will poll a device. With Ethernet communications that can be a lot of data very fast. To optimize this performance OPC specifies data reporting by exception, the exception being a data change. This is good as a large portion of the data that is contained in a PLC's I/O and memory is Configuration and Set Point information that changes very rarely. So why should we waste resources by updating the client application with the same data on every poll cycle. By subscribing a group the client is telling the server to only give it updates when the data changes.**

Next we need to set the parameters needed to add an item to the group so that we can get data updates from it. First we set our "ItemCount" parameter to 1 as we will only be adding 1 item in this example. You can add multiple items at one time to the group or you can add them as needed. We like to add them at the beginning of the process run since it can cause the server to pause its polling to add a new item. We then define the Item we are

adding. In KEPServerEX you would use the complete identifier which consists of the Channel Name, Device Name, and Item Name or Address separated by periods. Our item is Channel\_1.Device\_1.R0" which is an address in the "Simdemo" project mentioned earlier. You will notice that the item naming convention lets the server know exactly where to request the data from.

Next we are going to create an Item collection for our group and its "IsActive" property to "True" so that the item being added will be actively polled. The only thing left is to add the item to the Item collection.

**Note: In our example we do not have an error handling section but for live projects you will want to have a section to handle errors. When adding items the server will actually return error codes for successful completion as well as error codes for incompleteness. The error codes can indicate why the Add Item function failed. If you look at the Simple VB Example code that is shipped with the server you will see how this can be done.**

Add the code in Figure 3 to your Project Code window.

```
' General startup initialization
Private Sub Form_Load()
    'Create a new OPC Server object
    Set ConnectedOPCServer = New OPCServer
    ConnectedOPCServer.Connect "Kepware.KEPServerEX.V4"

    ' Add the group and set its update rate
    Set ConnectedServerGroups = ConnectedOPCServer.OPCGroups
    Set ConnectedGroup = ConnectedServerGroups.Add("Test1")

    ' Set the update rate for the group
    ConnectedGroup.UpdateRate = 500
    ' Subscribe the group so that you will be able to get the data change
    ' callbacks from the server
    ConnectedGroup.IsSubscribed = True

    ItemCount = 1
    OPCItemIDs(1) = "Channel_1.Device_1.R0"
    ClientHandles(1) = 1
    ' Establish a connection to the OPC item interface of the connected group
    Set OPCItemCollection = ConnectedGroup.OPCItems
    OPCItemCollection.DefaultIsActive = True
    OPCItemCollection.AddItem ItemCount, OPCItemIDs, ClientHandles, ItemServerHandles, ItemServerErrors

End Sub
```

Figure 3

## Group Data Change Event Code

As you may recall when we defined the group object we did so "With Events". One of the events is the Group Data change event. This is a call from the OPC Server to the client with data changes for the items that the group has asked for in its Item Collection. To do this three very important things had to occur when the client application connected to the server. First, the Group had to be made active, the default state is active. Second, the group had to be subscribed. Third, the item or item collection had to be made active. If this is done then you should get the data change events when they happen.

**Note: The item we are looking at in the Simdemo project is designed to increment every time it is polled. One you complete the following code you should be able to see this happen twice a second as that is the update rate we chose.**

You will have to manually type the Sub in Figure 4 into your Project Code window. We do not do anything fancy because we are only reading one item. If you were reading multiple items you would have to step through the ClientHandle array that is sent by the server so that you could update the proper item object values.

In our case we write the returned ItemValue to the text field property of our Text object.

Enter the code in Figure 4 to your Project Code window.

```
Sub ConnectedGroup_DataChange(ByVal TransactionID As Long, ByVal NumItems As Long, _
    ClientHandles() As Long, ItemValues() As Variant, Qualities() As Long, TimeStamps() As Date)
    ' We don't have error handling here since this is an event called from the OPC interface

    ' You can use the 'Clienthandles' array returned by the server to pull out the
    ' index number of the control to update and load the value. Since we only have
    ' one we do not worry about that.
    MyText.Text = ItemValues(1)

End Sub
```

Figure 4

## Form Unload or Exit Code

Ok, when you are done processing it is very important to remove all of your items and disconnect from the server so that it can properly re-allocate system resources that it is using and release those it no longer needs. Now if your project crashes obviously you cannot do this but if you are in control of closing it then you can. The two pieces of code in Figure 5 call the subs that remove the items and groups and disconnects from the server. You ask shy to subs, well you can close a form by unloading it or clicking on the exit button. So the first part is the code for the Click event of the button we added to the form earlier. The second part is for closing the form by clicking the form exit button in the top corner.

Enter the code in Figure 5 into your Project Code window.

```
Private Sub ExitExample_Click()
    'When you unload or close the form it is time to remove the Items, Group, and Server Connection
    Call Remove_Items
    Call Remove_Group
    Call Disconnect_Server
End
End Sub

Private Sub Form_Unload(Cancel As Integer)
    'When you unload or close the form it is time to remove the Items, Group, and Server Connection
    Call Remove_Items
    Call Remove_Group
    Call Disconnect_Server
End
End Sub
```

Figure 5

## Remove Items Code

Ok the first thing to do when closing out your project is to remove the items from the server project. We do this with the remove method of the Item collection object. It is essentially the reverse of the add item function. We start by defining a couple of variables and then setting their values as needed. We then execute the remove method.

**Note: There are two methods for removing items one is the Remove and the other is RemoveAll. I find that the Remove method is cleaner to use even though it might require more code to do.**

The last thing we do is release the item by setting it to nothing.

Enter the code in Figure 6 into your Project Code window.

```
Sub Remove_Items()  
Dim RemoveItemServerHandles(1) As Long  
Dim RemoveItemServerErrors() As Long  
  
'When you unload or close the form it is time to remove the Items, Group, and Server Connection  
  
'Remove the items  
RemoveItemServerHandles(1) = ItemServerHandles(1)  
OPCItemCollection.Remove 1, RemoveItemServerHandles, RemoveItemServerErrors  
'Release the item interface and all the resources to be freed  
Set OPCItemCollection = Nothing  
End Sub
```

**Figure 6**

### **Remove Groups Code**

Once you have removed the items you can remove the group by using the Group collections Remove method.

The last thing you do is to release the group and group collection objects by setting them to nothing.

Enter the code in Figure 7 into your Project Code window.

```
Sub Remove_Group()  
'Remove the group  
ConnectedServerGroups.Remove ("Test1")  
' Release the group interface and allow the server to cleanup the resources used  
Set ConnectedGroup = Nothing  
Set ConnectedServerGroups = Nothing  
  
End Sub
```

**Figure 7**

### **Disconnect Server Code**

Now you will use the Server Objects Disconnect method to disconnect from the server and then release the object by setting it to nothing.

Enter the code in Figure 8 into your Project Code window.

```
Sub Disconnect_Server()  
'Remove/Disconnect the server connecton  
ConnectedOPCServer.Disconnect  
' Release the old instance of the OPC Server object and allow the resources to be freed  
Set ConnectedOPCServer = Nothing  
End Sub
```

**Figure 8**

### **Run the Project**

Well if you entered the code properly you should be able to run the VB project and see the data change in your text box. If you had errors then you will need to debug them and try again. Figure 9 shows our project running with live data.

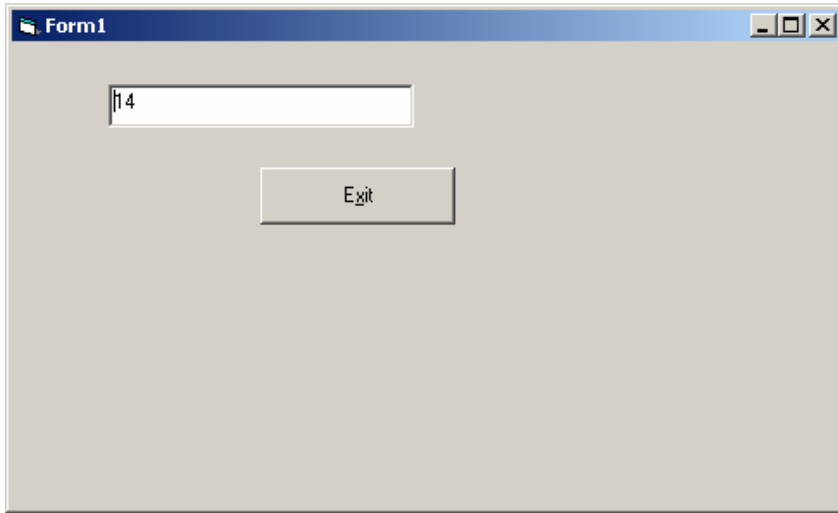


Figure 9

## Summary

Well you should now have a very basic understanding of how to create a VB Project that communicates to and OPC server. You may have noticed that you got a copy of the code for this project with the download so that you can compare it with your code. If you feel really adventurous you can look at the Simple and Complex VB projects that we install with our server.

If you have questions about the OPC portion of this VB project then please contact Kepware Technical support via e-mail at [Technical.Support@kepware.com](mailto:Technical.Support@kepware.com), via our Technical Support Feedback form on our website at [www.kepware.com](http://www.kepware.com), or via phone at 1-207-775-1660 x211.