



Visual Basic Grid Example

For all Kepware OPC servers

4/3/06 – v. 1.02

Introduction

This document describes an example of reading data from an OPC server and displaying that data in a grid in an OPC client application written in Visual Basic 6. This example is written for an intermediate-level VB programmer who is familiar with the minimum steps to connect a VB application to an OPC server and collect data. While reading this document, please refer to the accompanying example files:

- a VB project (Grid_Example.vbp), which can be compiled into a VB application;
- an OPC server project (Grid_Example.opf); and
- a text configuration file (“tags.txt”) for the VB application.

The OPC server project was made with KEPServerEX version 4.190.341 and must be opened in that version or higher. The principles described in this document apply to all versions of Kepware OPC servers, however.

Creating the VB Application

Install KEPServerEX

The first thing you will need to do, if you have not already done so, is to install KEPServerEX on your PC. 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 an example server project (.opf file) with 100 tags.

Start Visual Basic

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

Add Objects to the Form

For this example, we need to add three objects to the form.

The first object will be the grid itself. This is the MSFlexGrid control that is included with Visual Basic 6. The MSFlexGrid is found in the Toolbox, as shown below.



Figure 1

Once the MSFlexGrid is on the form, you can use the Properties Window in VB to modify the properties of the grid; for example, to show scrollbars. Please refer to the supplied VB project for example modifications.

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

The third object will be a timer, which is used in the shutdown process. Name the timer “tmrShutdown” in the Name property. The timer is not visible when the application is running.

Reference the OPC Automation Wrapper

There is no way for VB to access the server directly via OPC, so you will need to include a DLL in your project that will provide the software hooks you need. The KEPServerEX installation includes a generic OPC Automation wrapper. The file is located in the system32 folder of the operating system root directory. The file is named “OPCDAAuto.dll”.

In your project, click on Project | References... in the VB main menu to open the References dialog box. Then check the OPC DA Automation Wrapper checkbox. The version of the wrapper can be listed as 2.02 or higher.

Add the Code

The code in the project has the following sections:

- Project Declarations;
- Form Load;
- Group Data Change Event;
- Form Unload or Exit;
- Remove Group; and
- Disconnect Server.

Project Declarations Code

In the Declarations section, declare the global settings, constants, objects, and variables that you are going to use in this project.

In your project, open the form code window and enter the code in Figure 2 into the General Declarations portion of the code window.

```

Option Explicit
Option Base 1

Const intMaxRows As Long = 10
Const intMaxRowIdx As Long = intMaxRows - 1
Const intMaxCols As Long = 10
Const intMaxColIdx As Long = intMaxCols - 1
Const intMaxItems As Long = intMaxRows * intMaxCols
Const glbFile As String = ".\tags.txt"

Dim WithEvents ConnectedOPCServer As OPCServer
Dim ConnectedServerGroups As OPCGroups
Dim WithEvents ConnectedGroup As OPCGroup

' OPC Item related data
Dim OPCItemCollection As OPCItems
Dim OPCItemIDs(intMaxItems) As String
Dim ItemServerHandles(intMaxItems) As Long

```

Figure 2

Form Load Code

Now that you have defined the objects, it is time to start building the project. In the Form Load subroutine of the form, put the code needed to connect to the OPC server and establish the data polling.

After initializing the OPC connection, set the parameters needed to add an item to the group so that the application can get data updates from it. First, set the “ItemCount” parameter to 1, since only 1 item at a time will be added in this example.

Then, define the item ID of the item that is being added. In this example, read the item ID’s from a text file using a Do While loop.

Assign the ClientHandles systematically so that the DataChange event handler (discussed later) will be able to associate each ClientHandle with a location on the grid. In this example, assign the ClientHandles in sequence from 1 to 100.

Add each item to the item collection, then to a set of arrays for future reference.

Once all the items have been added, set the Group subscription property to “True.”

The last step in the Form Load code is to initialize the grid column widths.

Note: In this example, there is no error-handling section, but for projects to be used in production, you will want to have a section to handle errors.

Add the code in Figure 3 to your project code window.

```

' General startup initialization
Private Sub Form_Load()
Dim i As Integer

'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 OPCItemCollection = ConnectedGroup.OPCItems
OPCItemCollection.DefaultIsActive = True
ConnectedGroup.UpdateRate = 500

'Next we will read the text file and add the items.
*****

Dim ItemCount As Long
Dim OPCItemID(1) As String
Dim ClientHandle(1) As Long
Dim ItemServerHandle() As Long
Dim ItemServerError() As Long

Dim strTagID As String
Dim intAryIndex As Long

ItemCount = 1

intAryIndex = 1

Open glbFile For Input As #1
Do While ((Not EOF(1)) And (intAryIndex <= intMaxItems))

    Input #1, strTagID

' Establish a connection to the OPC item interface of the connected group
OPCItemID(1) = strTagID
ClientHandle(1) = intAryIndex
OPCItemCollection.AddItem ItemCount, OPCItemID, ClientHandle, ItemServerHandle, ItemServerError

OPCItemIDs(intAryIndex) = OPCItemID(1)
ClientHandles(intAryIndex) = ClientHandle(1)
ItemServerHandles(intAryIndex) = ItemServerHandle(1)

    intAryIndex = intAryIndex + 1

Loop
Close #1

' Subscribe the group so that you will be able to get the data change
' callbacks from the server
ConnectedGroup.IsSubscribed = True

'Initialize the grid column widths
For i = 0 To intMaxColIdx
    objGrid_Matrix.ColWidth(i) = 1600
Next

End Sub

```

Figure 3

Group Data Change Event Code

The Sub in Figure 4 is the group DataChange event handler. Put the following steps in it:

1. Start a loop to step through the ClientHandles array that is sent by the server. To determine the size of the array, use the NumItems value sent by the server.

For each item:

2. Convert the client handle for the current item to row and column coordinates, and make those coordinates the current ones by setting the Row and Col properties of the grid.
3. Update the grid's Text property with the value sent in the ItemValues() array.
4. Update the grid's CellBackColor property according to the quality sent in the Qualities() array.

Finally:

5. End the loop.

Enter the code in Figure 4 into 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

    Dim x As Integer

    For x = 1 To NumItems
        ' Use the 'Clienthandles' array returned by the server to pull out the
        ' row and column number of the grid cell to update and load the value.

        objGrid_Matrix.Row = Int((ClientHandles(x) - 1) / intMaxCols)
        objGrid_Matrix.Col = (ClientHandles(x) - 1) Mod intMaxCols
        objGrid_Matrix.Text = ItemValues(x)

        ' Check the Qualities for each item returned here. The
        ' quality field can contain bit field data which can provide specific
        ' error conditions. Normally, if everything is OK, then the quality will
        ' contain 0xC0.
        ' If the quality is good, we will set the cell background to vbWhite.
        ' If the quality is bad, then we will set the cell background color to red

        If Qualities(x) And &HC0 Then
            objGrid_Matrix.CellBackColor = vbWhite
        Else
            objGrid_Matrix.CellBackColor = vbRed
        End If

    Next x

End Sub
```

Figure 4

Form Unload or Exit Code

Define two event handler subs, ExitExample_Click and Form_Unload (Figure 5). These call the subs that remove the group and disconnect from the server. The ExitExample_Click routine is the code for the Click event of the button that you added to the form earlier. The Form_Unload routine is for when the user closes the form by clicking the form exit button in the top corner.

In each of the two event handlers, call a sub named OPC_Exit. The Form_Unload routine needs some additional code, which will be described later.

In the OPC_Exit sub, first disable all controls on the form to prevent the user from generating any more events during shutdown. Then, set the Group's IsActive property to False and begin a delay (200 ms in this example). This delay gives the server time to complete any outstanding onDataChange events. The delay is carried out by the timer control that you added to the form earlier.

Add a `tmrShutdown_Timer` event handler, which runs when the delay expires. In this routine, first disable the timer to prevent further timer events from occurring. Then, unsubscribe the group and call the `Remove_Group` and `Disconnect_Server` routines. Finally, add the `End` statement, which completes unloading the form and stopping execution of the `.exe`.

The additional code in the `Form_Unload` routine is to set the `Cancel` parameter. This way, the form will not be unloaded until the code started by `OPC_Exit` has had time to complete. The `End` statement in the `tmrShutdown_Timer` routine will finish unloading the form.

Enter the code in Figure 5 into the project code window.

```
Private Sub OPC_Exit()  
    'When you unload or close the form it is time to remove the Items, Group, and Server Connection  
  
    Dim AControl As Control  
  
    ' Disable all controls  
    For Each AControl In View.Controls  
        AControl.Enabled = False  
    Next AControl  
  
    ' Make sure no data change callbacks will occur during shutdown  
    ConnectedGroup.IsActive = False  
    tmrShutdown.Interval = 200  
    tmrShutdown.Enabled = True  
  
    ' Now the application will wait for the timer interval to expire.  
    ' This delay is to make sure that all pending onDataChange()'s have  
    ' completed before this application attempts to remove the group.  
  
End Sub  
  
Private Sub tmrShutdown_Timer()  
    ' Finish shutting down the application.  
  
    tmrShutdown.Enabled = False  
  
    ConnectedGroup.IsSubscribed = False  
  
    Call Remove_Group  
    Call Disconnect_Server  
  
End  
  
End Sub  
  
Private Sub ExitExample_Click()  
    Call OPC_Exit  
End Sub  
  
Private Sub Form_Unload(Cancel As Integer)  
    Call OPC_Exit  
  
    Cancel = 1  
    ' Let the shutdown timer run.  
    ' The timer event handler (which runs when the timer expires) will  
    ' finish closing this application.  
  
End Sub
```

Figure 5

Remove Group Code

You can remove the group and the items by using the Group collection's RemoveAll method.

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

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

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

Figure 6

Disconnect Server Code

Now you will use the Server object's Disconnect method to disconnect from the server. Then, release the object by setting it to nothing.

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

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

Figure 7

Run the Project

Now you can run the VB project and see the data updated in the grid. Figure 8 shows the project running with live data.

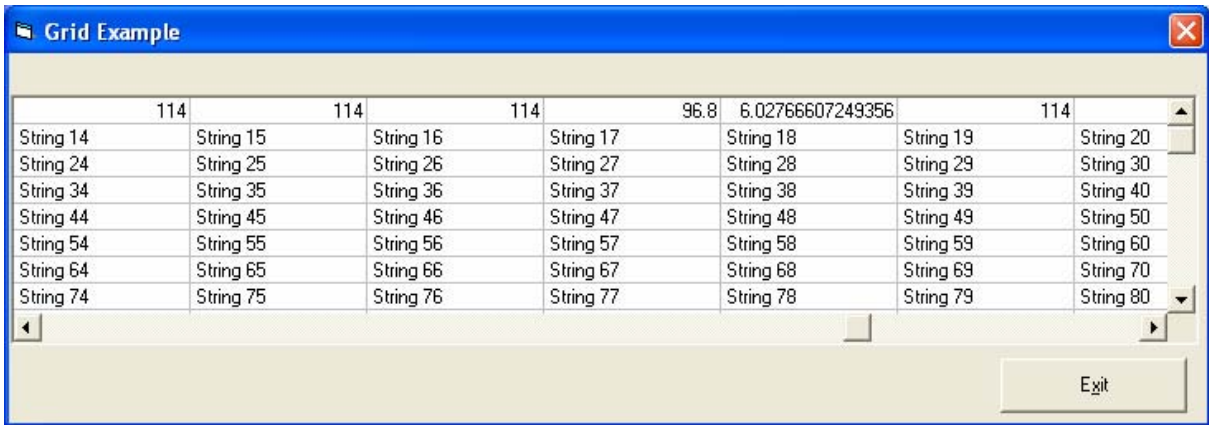


Figure 8

Distributing the VB Application

This grid application requires the file `msflxgrd.ocx`, which is supplied with Visual Basic. To distribute the application to machines other than the development machine, you must generate a package which includes the `.ocx` file as well as the `.exe` itself. To do this, use the Package and Deployment add-in that comes with Visual Basic.

Start the Package and Deployment Add-in

In the Visual Basic menu bar, go to Add-Ins.

If you don't see an option for Package and Deployment, then:

1. Choose the Add-In Manager option from the menu.
2. In the Add-In Manager window, click on the name of the Package and Deployment add-in.
3. Check the Loaded/Unloaded and Load on Startup checkboxes, as shown below.

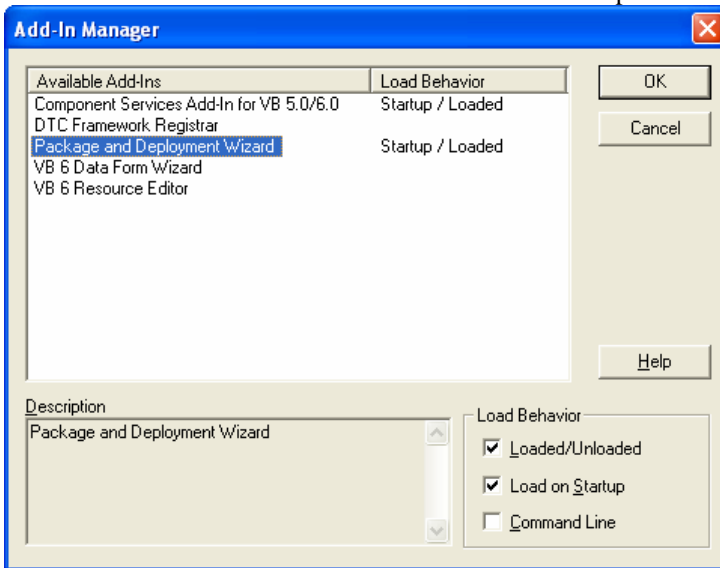


Figure 9

4. OK the Add-In Manager.

Now that the Package and Deployment add-in is listed under Add-Ins, select the menu option for it. This will start the add-in.

Select Package and Deployment Options

Once the Package and Deployment window appears, click on Package, as shown in the following figure.

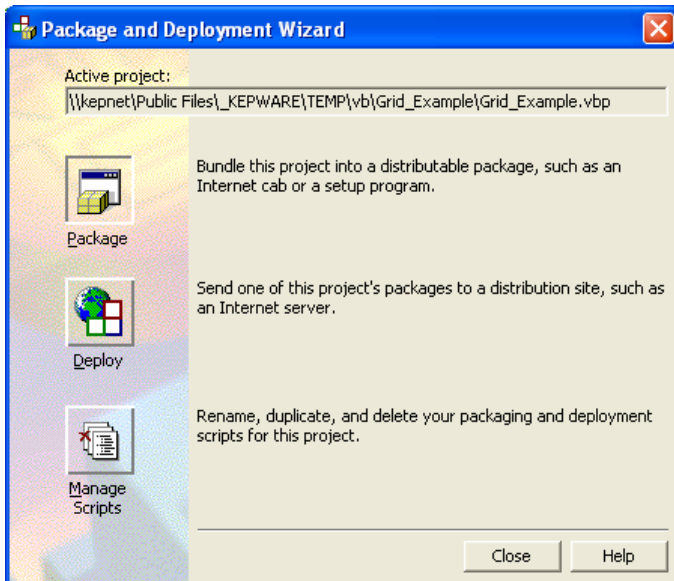


Figure 10

If you are asked whether to recompile, choose Yes.

The add-in will do some processing and then ask you what packaging script to use and what type of package to create (screens not shown). You can accept the defaults, clicking on Next at each step.

Then, as shown below, select the folder where the installation package will be generated. Click Next to continue.

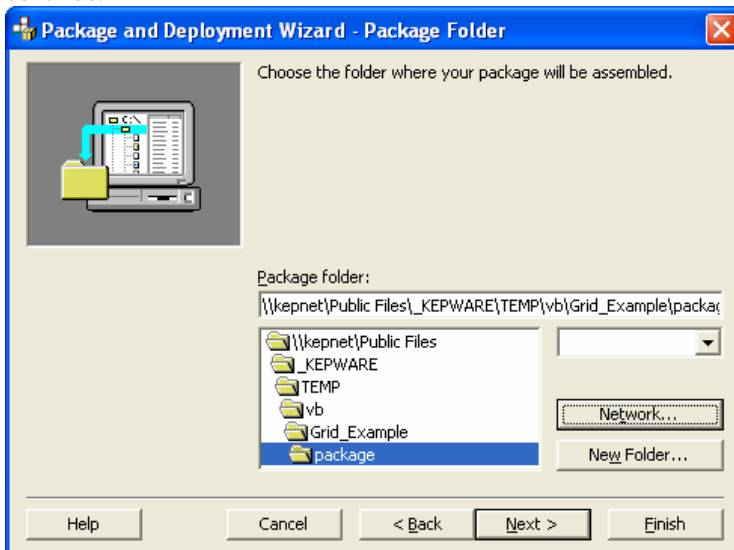


Figure 11

The next screen will show you what files will be included in the package. Notice that msflxgrd.ocx is automatically put in the list. Click Next to move on.

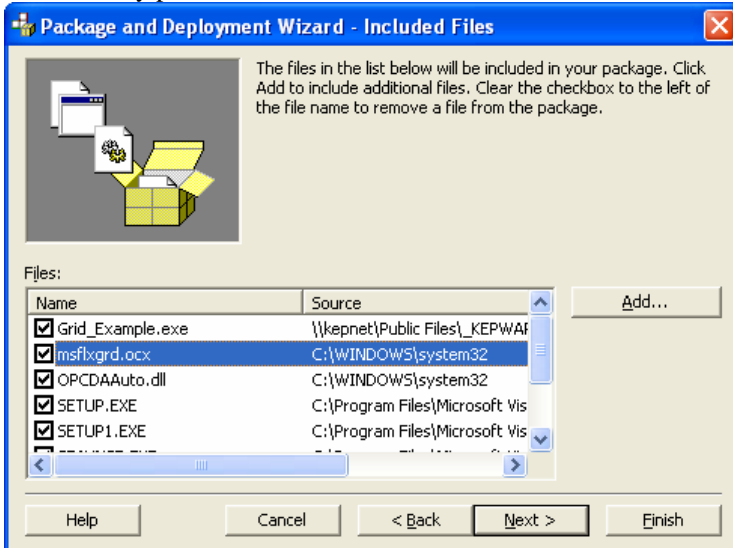


Figure 12

Then, you will be asked whether the package will go into a single .cab file or multiple .cab's (screen not shown). You can leave this at the default setting and click Next.

In the screen after that (not shown), you can enter an installation title and then click Next to proceed.

Then, in the Start Menu Items screen, specify where in the Windows Start menu the program shortcut will appear.

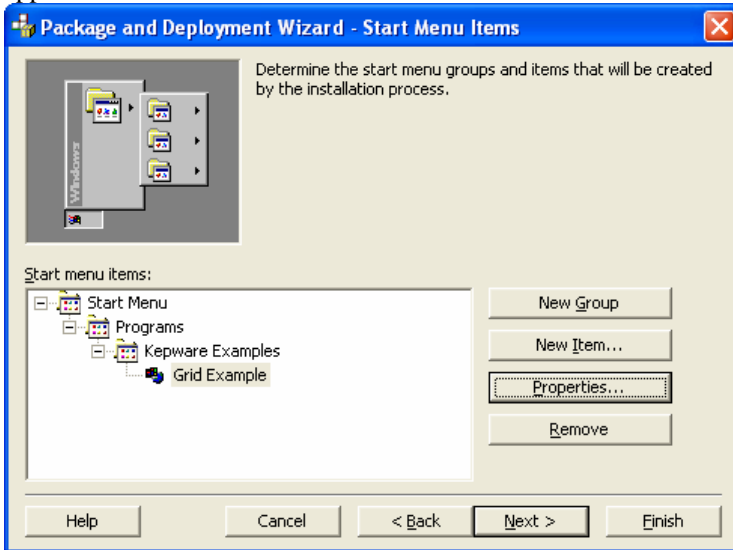


Figure 13

Click Next to bring up the Install Locations screen (not shown). You can accept the defaults and click Next.

Then you will be asked if you want the .exe to be installed as shared (screen not shown). Accept the default (not shared) and click Next.

On the Finished! screen (not shown), you can accept the default script name and click Finish. Then, if you are asked whether to overwrite the script, click Yes.

Finally, after the add-in does some more processing, you will be presented with a report (not shown). Click Close to dismiss the report, then Close again to exit the Packaging and Deployment add-in.

If you look in Windows Explorer at the folder you chose for generating the package, you will see something like the following:

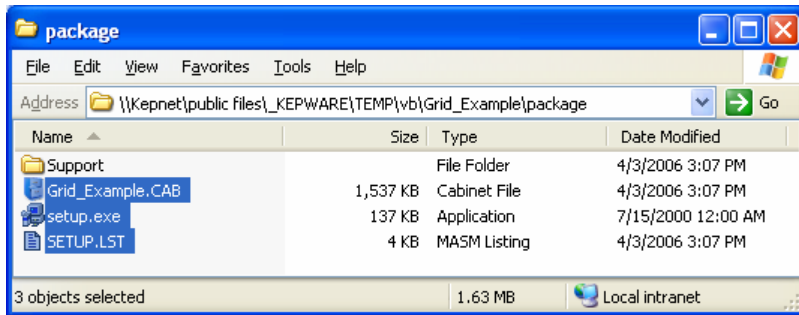


Figure 14

Only the .cab, setup.lst, and setup.exe files are part of the package. If you copy them to another PC and then run setup.exe, your program will be installed into the Start menu on that PC.

In this example, you also need to make sure that “tags.txt” is copied to the same directory as the VB .exe, and that the OPC server on that machine is set to start up with the correct project file.

Summary

You should now have an understanding of how to make a VB project with a grid that displays data from an OPC server. You can also look at the Simple and Complex VB projects that are installed with our server.

If you have questions about the OPC portion of this VB project, please contact Kepware Technical Support via e-mail at Technical.Support@kepware.com, via our technical support feedback form on our website at www.kepware.com, or via phone at 1-207-775-1660 x211.