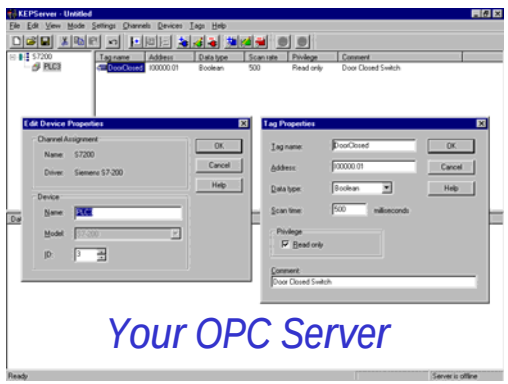
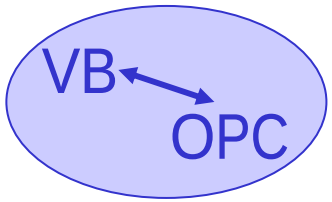
# Using Visual Basic As An OPC Client

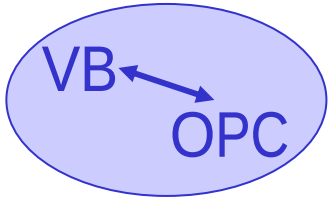

*John Weber*
*President & Founder*
*Software Toolbox, Inc.*
*jweber@softwaretoolbox.com*
*website: http://softwaretoolbox.com*

*Presentation Updated 3/2001*



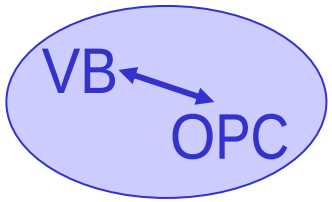

# Agenda

- OPC - Review of the Basics
- OPC - Terminology
- OPC & VB 101 - Automation Wrapper & Object Model
- Building Your VB Client - Read data - 7 easy steps with code from start to finish
- Housekeeping
- Handles Handles Everywhere - how the server and client correlate the data each is managing in your program
- Writing Data
- Future ideas to consider
- New developments since original presentation in 10/99
- Resources for future learning



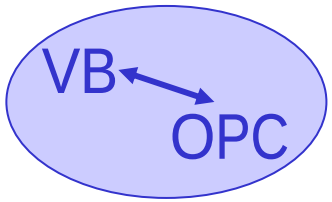

# OPC - Review of the Basics

- OPC is based on the Microsoft Component Object Model (COM)
- OPC stands for OLE for Process Control
- OPC is managed by the independent OPC Foundation (www.opcfoundation.org)
- There are over 200 member companies supporting the OPC specification
- OPC defines a standard method for connecting automation application software



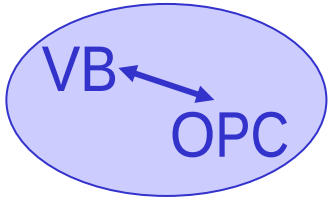

# OPC and VB 101 - Terminology

- **OPC Server** - a piece of software supporting the OPC specification - an OPC Data Access Server is a driver that handles connectivity to PLCs or other automation hardware
- **OPC Item** - A single tag or data point managed by the OPC server - i.e. one data point in a PLC
- **OPC Group** - a user defined grouping of OPC items. Created by the OPC client as a way of organizing data items around logical process areas or equipment.
- **Collection** - a grouping of objects of the same data type - for example, each OPC server has an OPC items collection containing one software object for each OPC item.



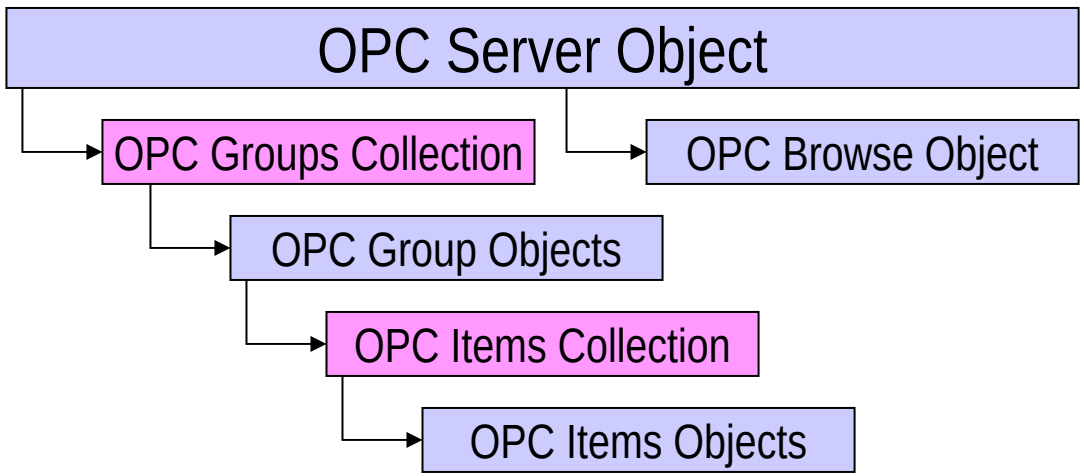

# OPC and VB 101

- Visual Basic supports COM
- COM implementations from Visual Basic use what is called an “Automation” interface
- The OPC Foundation supplies the source code to an “Automation Wrapper” DLL - most OPC vendors provide a compiled version
- Automation Wrapper connects VB to OPC

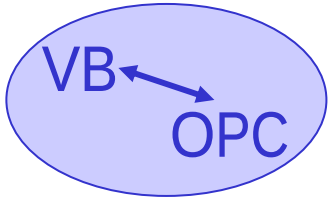



# OPC and VB 101

- Automation Wrapper DLL lets VB access OPC Servers and their underlying Groups and Items




**Object model for the Automation Wrapper -** the wrapper lets the VB user connect to an OPC server using standard Object.property and Object.method syntaxes

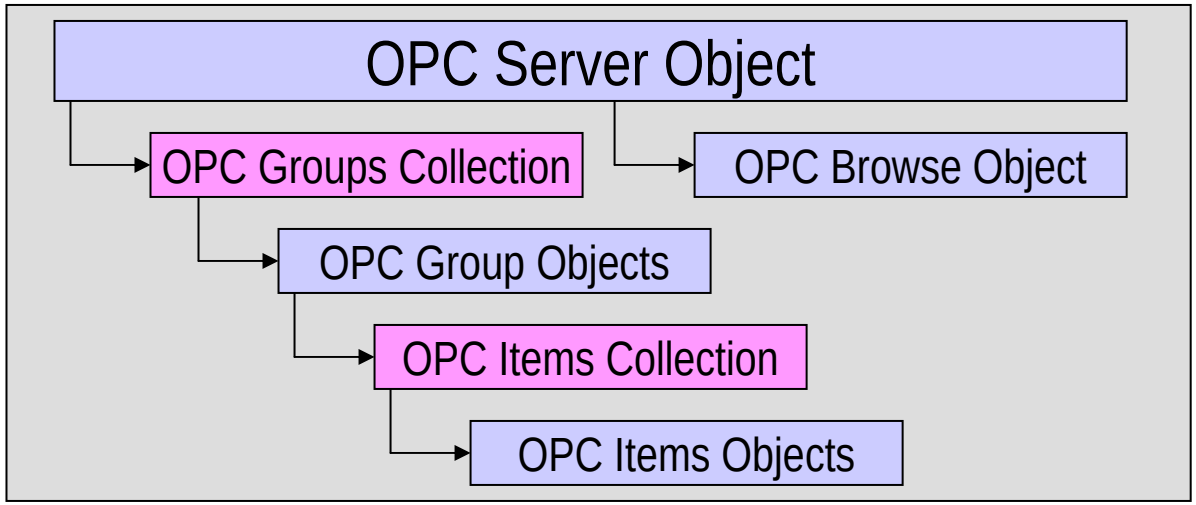


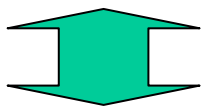

# OPC and VB 101

Your VB Program with the Automation Wrapper object included in it

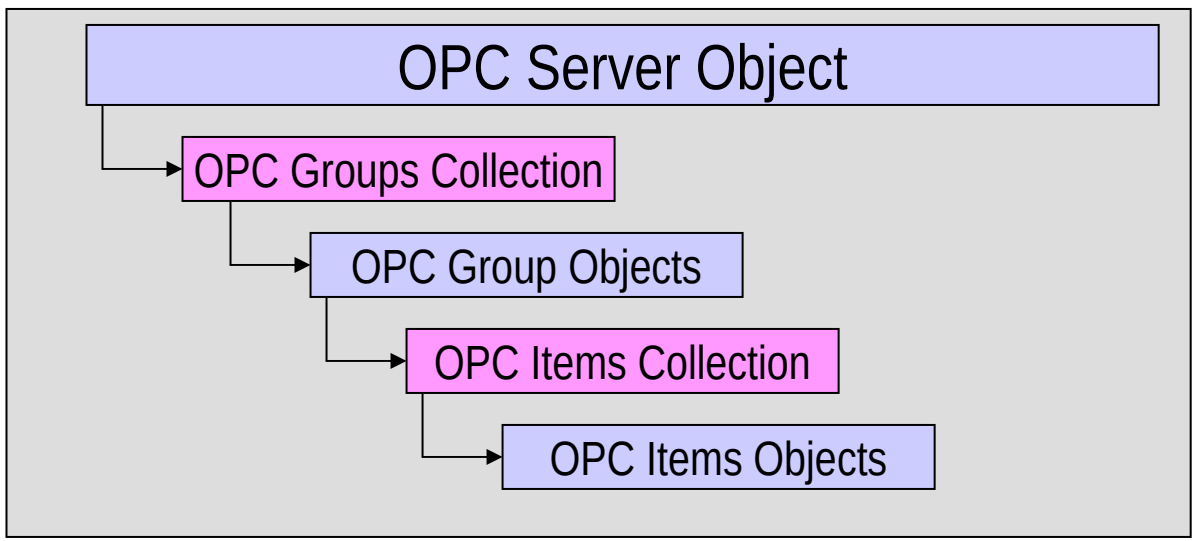
The Automation Wrapper connects to the OPC server and creates the groups and items in the server and gives you references to them in your VB program in an Object model that mirrors that of the server

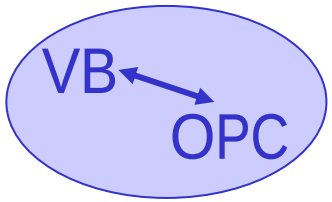
Your OPC Server and it's object model



COM/DCOM






# OPC & VB - Making The Connections

Connect to the OPC Server

Add Group(s)

Add Item(s) to Group(s)

Activate Groups & Items

Data Change Events Fire

User Program Handles Data - Log, Display, Graph, etc

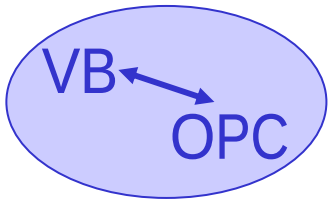
*Optional - your program can add/remove items and groups and activate/deactive items at runtime based on user defined conditions*

De-Activate Groups & Items

Remove Item(s) from Group(s)

Remove Group(s)

Disconnect from the OPC Server

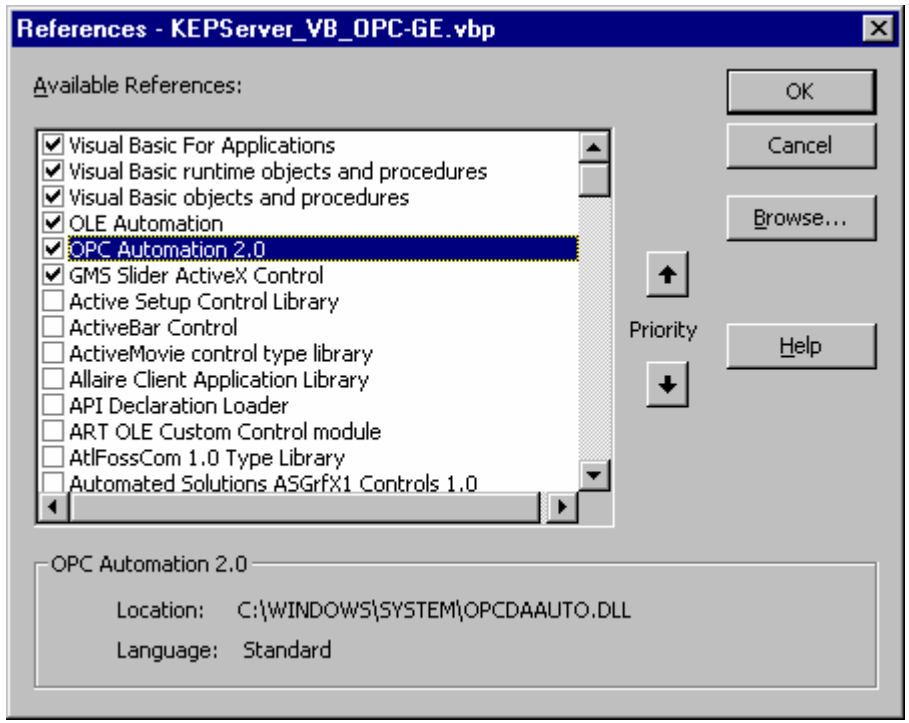


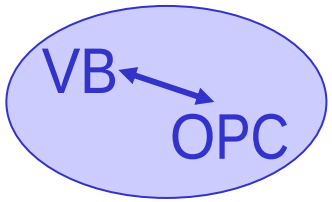

# Tools You'll Need To Build Your Application

- If you want to build an OPC client in VB and test it, you'll need the following tools:
  - Visual Basic 5 or 6 running on Windows 95, 98, or NT any VB edition will do the job
  - An OPC Server
  - The OPC Automation Wrapper
- You can get the last two items including a sample test OPC server at softwaretoolbox.com (see last slide for exact link). There is no charge for the Automation Wrapper and sample OPC demonstration servers are free also.

# Building Your VB Client - Step 1

- Install the OPC Automation Wrapper DLL on your PC
- Start a new VB project
- In VB, click on Project --> References on the VB menu bar
- The OPC Automation Wrapper appears on the dialog as “OPC Automation 2.0” - select it as shown here.





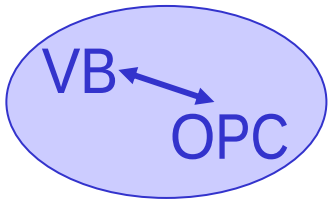

# Building Your VB Client - Step 2

- First you need to declare some variables in the VB code window in the General Declarations area as shown here

```
Option Explicit
Option Base 1 'Makes all arrays start with an index of 1

Dim WithEvents AnOPCServer As OPCServer
Dim WithEvents ConnectedOPCServer As OPCServer
Dim ConnectedServerGroup As OPCGroups
Dim WithEvents ConnectedGroup As OPCGroup

Dim OPCItemCollection As OPCItems
Dim ItemCount As Long
Dim OPCItemIDs(10) As String
Dim ItemServerHandles() As Long
Dim ItemServerErrors() As Long
Dim ClientHandles(10) As Long
```

*These lines create objects you will use to manage your OPC server connection and a group - you could add more than one group if you wanted to*

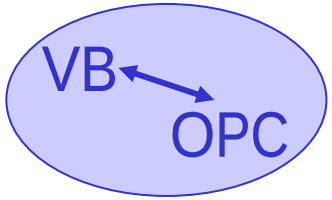
*These lines create objects you will use to manage your OPC Items - we are setting up our sample to read 10 items - you setup as many as you need*




# Building Your VB Client - Step 3

- If you want your VB project to connect to your server at startup, use the name of your server (ask your server vendor) and enter the following code in the Form Load subroutine for your project - our server name is “KepServer”

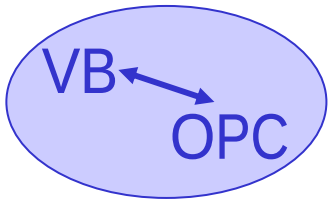
```
Dim ConnectedServerName As String

'Create a new OPC Server object
Set ConnectedOPCServer = New OPCServer
'Load the selected server name to start the interface
ConnectedServerName = “Developer - Enter Your OPC Server Name Here in quotes”
'Attempt to connect with the server (Local only in this example)
ConnectedOPCServer.Connect (ConnectedServerName)
```




# Building Your VB Client - Step 4

- Next, you'll go ahead and add a group right after you get your connection to the OPC server. Enter this code right after your code for connecting to the server in the Form Load subroutine of your project

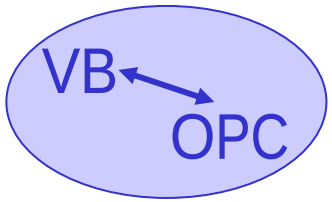
```
'Prepare to add a group to the current OPC Server
' Get the group interface from the server object
Set ConnectedServerGroup = ConnectedOPCServer.OPCGroups

' Set the desire active state for the group
ConnectedServerGroup.DefaultGroupIsActive = True

'Set the desired percent deadband - enter an integer from 0 to 100 for the deadband
ConnectedServerGroup.DefaultGroupDeadband = 0

' Add the group and set its update rate - enter whatever group name you want in place of "DataGroup1"
Set ConnectedGroup = ConnectedServerGroup.Add("DataGroup1")

' Set the update rate for the group - enter an long integer value representing the millisecond group update rate
ConnectedGroup.UpdateRate = 500

‘ The following line is crucial -- without it you won't be subscribed to the server and DataChange events will not fire!
ConnectedGroup.IsSubscribed = True
```




# Building Your VB Client - Step 5

- Next you'll go ahead and add some items. This code follows right after the group add code in the Form Load subroutine
- For the item names, enter valid item names for your OPC server. Refer to your OPC server's documentation for valid item naming conventions. We are going to be reading some data from a GE PLC here so we will use their memory conventions. The OPC server we are using uses the syntax "itemname@updaterate"

```
ItemCount = 4
Dim i As Integer
For i = 0 To 3
   'This line builds a string like "GE9030.R1@10" - a valid item name for the OPC server we are using
   OPCItemIDs(i + 1) = "GE9030.R" & (I + 1) & "@10"
   ClientHandles(i + 1) = I                    ' Sets a reference pointer number for this point
   OPCItemActiveState(i).Value = 1             ' Tells the server we want this item to be active
Next I
Set OPCItemCollection = ConnectedGroup.OPCItems    'Gets an items collection from the current Group
OPCItemCollection.DefaultIsActive = True           'Sets the items collection to active
'This line adds the items we've chosen to the items collection and in turn to the group in the OPC Server
OPCItemCollection.AddItems ItemCount, OPCItemIDs, ClientHandles, ItemServerHandles, ItemServerErrors
```

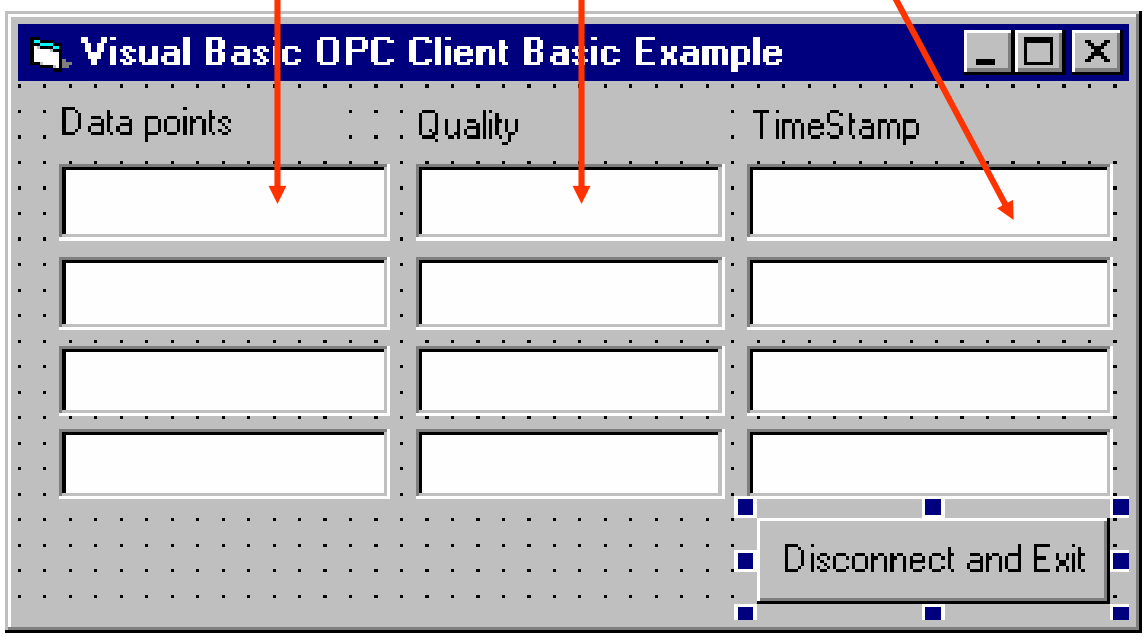


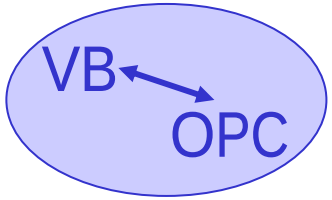

# Building Your VB Client - Step 6

- Now, assuming you have all valid item names and your OPC server is ready to go, all you need to do is add code to react to the DataChange events that your OPC server will fire back to the Automation DLL, which in turn will fire an event in VB for you
- In your program, you created an object variable called “ConnectedGroup” in step 2 -- this object will fire an event called “DataChange” for anytime one of the items in your group has new data for you
- The DataChange event tells you how many items changed, gives you back the client handles so you know which ones changed, and the data, quality, and timestamp information

# Building Your VB Client - Step 6

- Go ahead and build the objects on your form to display your data.
- For our form, we built one with 3 arrays of text boxes -- one each for the data, quality, and timestamp information. We named them txtData, txtQuality, txtTimeStamp respectively.






# Building Your VB Client - Step 6

• Now in the VB Code window, use the combo boxes at the top to go to the ConnectedGroup object's DataChange event subroutine -- enter this code to update your text boxes.

• Note this code assumes you have named your text boxes as we have in our example and built an array of text boxes as we have in our form. If you named your text boxes differently, then you will need to adjust this code accordingly.

• The variables ItemValues, Qualities, ClientHandles, and TimeStamps are all variables passed to you by the Automation Wrapper DLL and your Connected Group object when the DataChange event fires

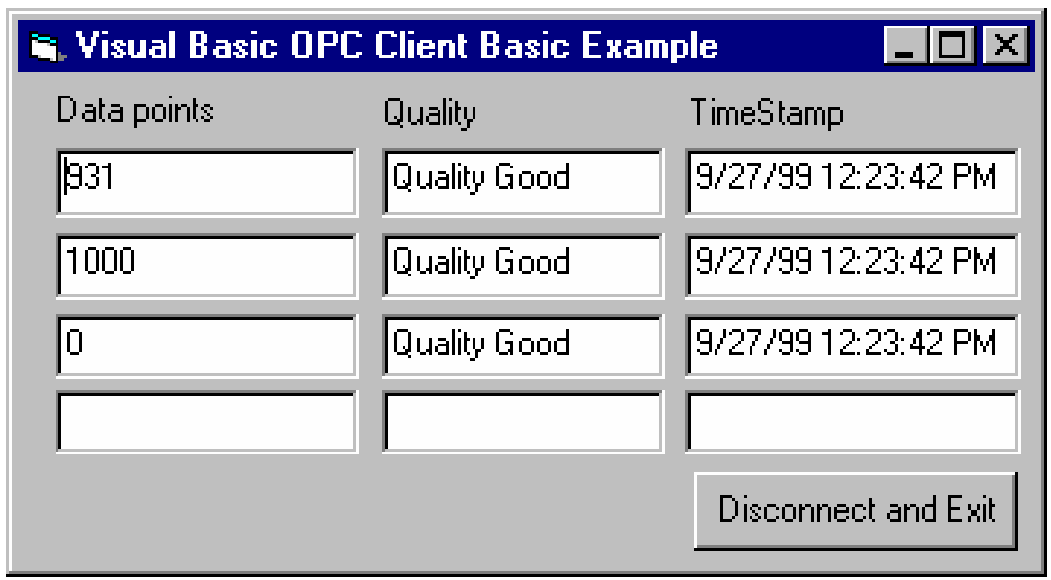
```
Dim i As Integer
  For i = 1 To NumItems
    txtData(ClientHandles(i)).Text = ItemValues(i)
    If Qualities(i) And &HC0 Then
      txtQuality(ClientHandles(i)).Text = "Quality Good"
    Else
      txtQuality(ClientHandles(i)).Text = "Quality Bad"
    End If
    txtTimeStamp(ClientHandles(i)).Text = TimeStamps(i)
  Next i
```

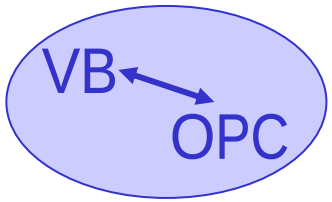
*The Qualities data array can contain a variety of very hardware specific data about why communications failed - a value of Hex C0 is returned when everything is OK*

*Timestamps are returned in GMT or Universal Time*

# Building Your VB Client - Step 7

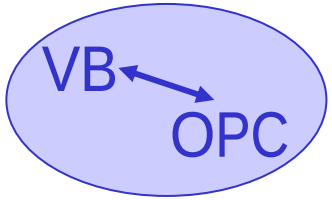
- Now if you have not done so, save your project and form in VB
- Run your project. If you have specified a valid OPC server and Item names and your OPC server is running and ready, then when you run your project, it should immediately start updating the text boxes on your form






# If things aren't working . . .

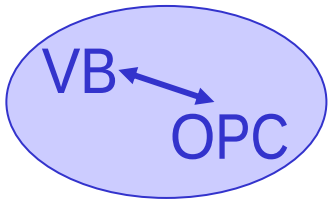
- If you do not have communications, check the following
  - Do you have the OPC server name specified correctly in the form load routine ? Check with your OPC server vendor for the right name to use
  - Are you using valid item naming syntaxes for your pariticular OPC server?
  - Do you have the cabling and hardware setup right between your OPC server and your hardware?




# Housekeeping

- This program was a very basic one -- in reality you need to add at least one more section of code in order to make sure you clean up your connections to the OPC server before you exit your program.
- Otherwise the OPC server will still think you are connected and hold open memory to service you that it could otherwise release.
- Enter the code on the following slide in the Click ( ) event on the “Disconnect and Exit” command button you put on your form
- This code makes sure you remove the item, items collection, group, and groups collections from the server then disconnect from the server upon exit.

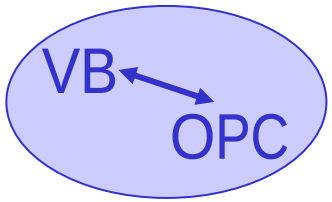
*Good memory management dictates that you “undo” everything that you “do”*




# Housekeeping - Exit & Cleanup Code

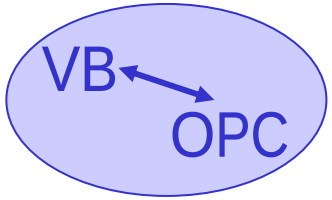
```
'========== First Remove All of Our Items =============='
    ItemCount = 1
    ' Provide an array to contain the ItemServerHandles of the item
    ' we intend to remove
    Dim RemoveItemServerHandles(10) As Long
    Dim RemoveItemServerErrors() As Long
    ' Get the Servers handle for the desired items.  The server handles
    ' were returned in add item subroutine.  In this case we need to get
    ' only the handles for item that are valid.
    Dim i As Integer
    For i = 1 To 3 ' We have 3 data points we are reading
       ' In this example if the ItemServerHandle is non zero it is valid
       If ItemServerHandles(i) <> 0 Then
          RemoveItemServerHandles(ItemCount) = ItemServerHandles(i)
          ItemCount = ItemCount + 1
       End If
    Next i
    ' Item count is 1 greater than it needs to be at this point
    ItemCount = ItemCount - 1
    ' Invoke the Remove Item operation.  Remember this call will
    ' wait until completion
    OPCItemCollection.Remove ItemCount, RemoveItemServerHandles, RemoveItemServerErrors
    ' Clear the ItemServerHandles and turn off the controls for interacting
    ' with the OPC items on the form.
    For i = 0 To 2
       ItemServerHandles(i + 1) = 0 'Mark the handle as empty
    Next i
    Set OPCItemCollection = Nothing
```

```
'======= Now remove the group ===========
   ' Remove the group from the server
   ConnectedServerGroup.Remove (Groupname)
   ' Release the group interface and allow the server to cleanup the resources
used
   Set ConnectedServerGroup = Nothing
   Set ConnectedGroup = Nothing

'======= Now Disconnect from the OPC Server ===========
   ConnectedOPCServer.Disconnect
   ' Release the old instance of the OPC Server object and allow the
resources
   ' to be freed
   Set ConnectedOPCServer = Nothing

'====== Now exit the program =============
   End
```




# Handles Handles Everywhere

- In the preceding program, you probably saw that we used the ClientHandles and ItemServerHandles variables a lot. Just what are these “handles”
- A handle is an identifier assigned in software to point to a memory location or data point.
- When you connect to an OPC Server and create items, the OPC Server assigns a handle for each item - a large number.
- Likewise, when your OPC client connects to the server, it needs to have a handle to uniquely identify each item.
- Handles as numbers are much easier for code to deal with quickly and efficiently behind the scenes than long names.
- Refer to Step 5 -- when you added items, you set your handles for the client side in the array variable ClientHandles ( )
- When you invoked the .AddItems method, you passed the server your list of ClientHandles( ) -- the OPC server returns a list of the corresponding handles it assigns in the array variable ItemServerHandles( )
- You use the ItemServerHandles() to tell the server which item you are interested in for future synchronous read and write operations on a single item.




# Handles Handles Everywhere

- The following picture should help you understand what handles look like inside the program.
- The numbers used here are actual numbers that were assigned in running the program created by this tutorial.
- Your ItemServerHandles will be different as each server on each PC assigns handles differently to assure uniqueness.

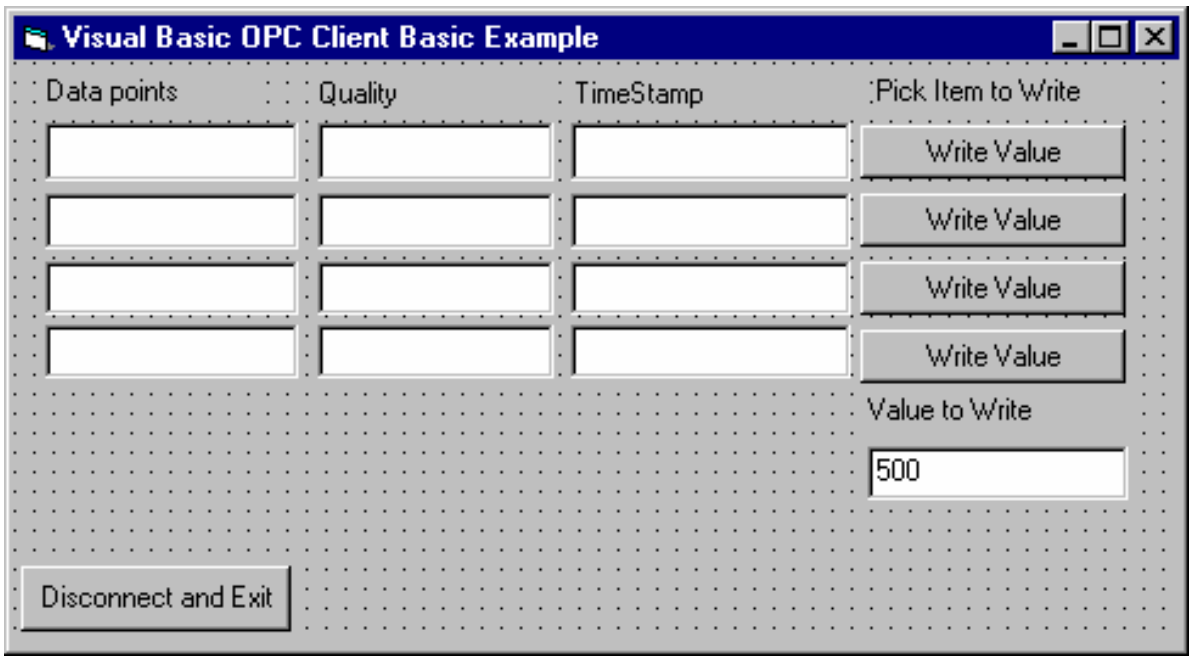
| Array Index | ClientHandles( ) | ItemServerHandles( ) |
|---|---|---|
| 1 | 0 | 38145856 |
| 2 | 1 | 38145968 |
| 3 | 2 | 38146080 |
| 4 | 3 | 38146192 |

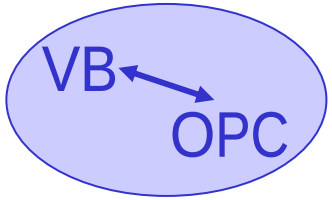
Assigned by your program and passed to the OPC Server when you call the .AddItems method. Should be numbers that are useful to you in your code in working with and manipulating the data

Assigned by the OPC Server and passed back to your program by the Automation Wrapper DLL when the AddItems method completes and returns

# Writing a Value - Step 1

- Of course now that you are reading data, you want to write some data. Add a text boxes and an array of command buttons to your form as shown below. Name your text boxe txtValueToWrite and the command button array cmdWriteValue. For simplicity, we will stick to just writing items that you already subscribed to when we loaded the form.

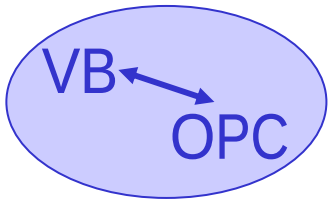





# Writing a Value - Step 2

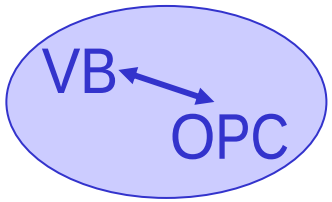
- When you click on the write command buttons, the event handler in VB hands you an index from 0 to 3 which matches up to the items that we added when the form loaded. The following code performs an OPC Synchronous Write to the chosen item - place this code in the cmdWriteClick event in your VB program

```
' Write only 1 item this time
ItemCount = 1
' Create some local scope variables to hold the value to be sent.
' These arrays could just as easily contain all of the item we have added.
Dim SyncItemValues(1) As Variant
Dim SyncItemServerHandles(1) As Long
Dim SyncItemServerErrors() As Long

' Get the Servers handle for the desired item.  The server handles
' were returned when we loaded the form. We use the index of the command button clicked to tell us which item to change
SyncItemServerHandles(1) = ItemServerHandles(Index + 1)

' Load the value to be written from our text box
SyncItemValues(1) = Val(txtValueToWrite.Text)

' Invoke the SyncWrite operation.  Remember this call will wait until completion
ConnectedGroup.SyncWrite ItemCount, SyncItemServerHandles, SyncItemValues, SyncItemServerErrors
```



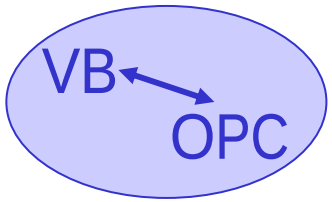

# Writing a Value - Step 3

- Run your program. Assuming of course that your PLC or device will let you write the points you are reading, you should now be able to enter a value in the text box and write a data point and see it change in the display where you were reading data.
- Words of wisdom on writing
  - You can't write to an item that you haven't added to the items collection.
  - You don't have to be reading an item in order to write it -- but it must be already added to the current group by your client program
  - Synchronous writes will take control and execute with high priority
  - Asynchronous writes are available for lower priority writes -- see the OPC Automation Specification for syntax for Asynch Writes



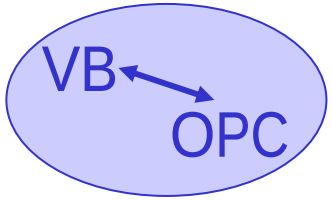

# From Here . . .

- From here you would want to build in some error handling. This sample assumes everything works and does not trap errors
- Download more samples (see last slide for links) for sample code including error handling
- Once you have the data back into your VB program (Data Change event), you can pass the values to any graphical ActiveX display object on the same form or a different form




# From Here . . .

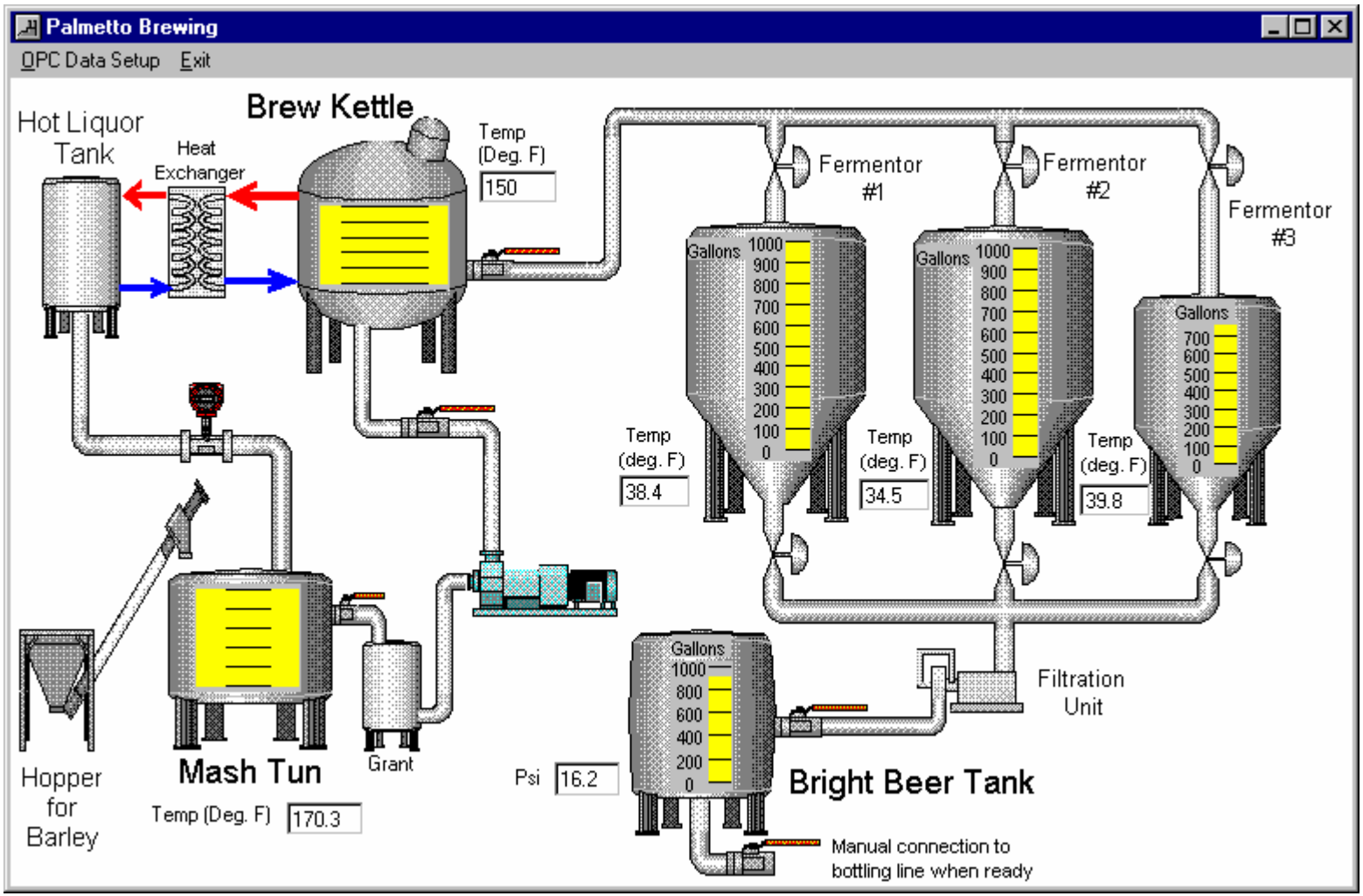
- Other ideas for thought . . .
  - VB can read or write any database type
  - Use VB to read your list of items and scan rates from a database upon form load and get all the points setup at the start. Add points by adding them to the database rather than modifying your code
  - Need to log data ? Use the Data Change event and VB's database tools to write selected datapoints to a database for historical storage

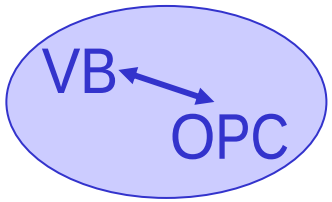



# New developments since original presentation created . . .

- www.OPCActiveX.com -
  - ActiveX control that encapsulates all the details for you
  - Read data into VB with no code needed
  - Write data with one line of code
  - Built in tool to browse remote servers and tags
  - Robust DCOM support
  - Tested with 40+ commercial OPC servers
- You can build yourself for free, or license existing technology at low prices - your choice

# The Possibilities . . . .






# OPC Resources

- OPC Foundation - www.opcfoundation.org
- Samples - softwaretoolbox.com/isaexpo99
  - The program created by this document
  - A more advanced program
  - Sample OPC Servers
  - Automation Wrapper DLL install set
  - This document in Powerpoint (*.ppt) format
  - OPC Automation 2.0 specification document
- Commercially supported tool to save code writing - www.OPCActivex.com
- Contact the presenter - email to jweber@softwaretoolbox.com