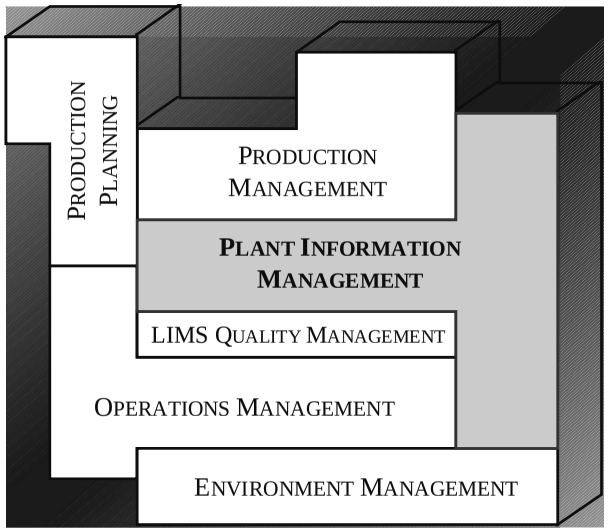
PHD Data in Web Displays



Lesson Objective

Objectives

- Create a simple web page using Active Server Page (ASP) scripting to access PHD data.
- Enable a VB application containing Visual PHD ActiveX controls to be viewed in a web browser.
- Recognize the tradeoffs of thin vs fat client web pages to access PHD data.

Topics

- Thin Client vs Fat Client Web Pages
- Active Server Pages (ASP)
- Simple ASP Example
- Hands-on Exercises - Thin Client
  - Access the current value of a hardcoded tag
  - Access the current value of a user-entered tag
  - Access the previous hour's data for a hardcoded tag
- Software Requirements
- Scripts to Test Installation
- How Visual PHD Works
- ActiveX Controls vs OLE Server
- Hands-on Exercise - Fat Client
  - VB application containing Visual PHD ActiveX Controls

References:
Online document:
VisualPHD Internet User Guide
CD document:
VisualPHD User Guide, PIM2201

Thin Client vs Fat Client Web Pages

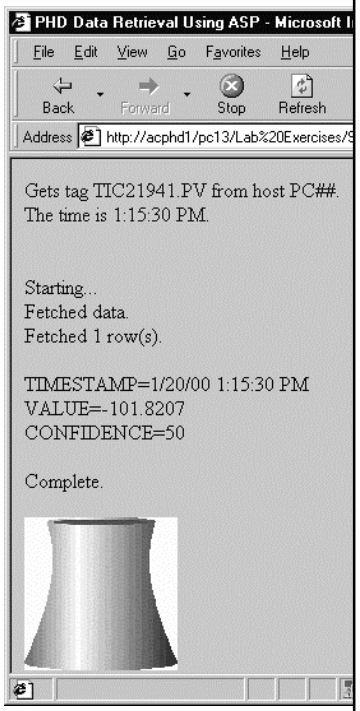
Web page scripting executes either on the client machine or on the web server machine.

When PHD retrieval is in…	Then.....
Server-Side Script	• The web page is a *Thin Client*. • VisualPHD and Oracle client must be installed on the web server only. • The web server does the PHD data retrieval, then sends the results to the client. • The client requires only the browser software. • Server-side scripts must handle communication: security, connection, and data retrieval.
Client-Side Script	• The web page is a *Fat Client*. • VisualPHD and Oracle client must be installed on the clients. • The client does the PHD data retrieval. • The security model is simpler, because the client is in full control of the security.

Active Server Pages (ASP)

- ASP files are HTML that has been filtered through the ASP processor. ASP files have the `.asp` extension.
- The web server runs files with the .asp extension through a preprocessor that processes the server-side (asp) scripting and generates the HTML that gets sent to the client's web browser.
- Files with other extensions (such as .HTM) get sent to the browser directly.
- An ASP file may contain HTML (client-side) and ASP (server-side) scripting.
- You designate ASP scripting with <% at the beginning and %> at the end.
- ASP uses Vbscript, a subset of VB. You lose some typing on variables (there is no 'dim as', just 'dim' - there is no 'as' anything in VBscript). Some of the VB commands are removed. Vbscript is upwardly compatible with VB.
- When you request a web page, server-side scripts execute first, then client-side scripts.
- When viewing the source of an ASP web page at the client side, you cannot see the server-side scripting.

Simple ASP Example




```
<html>
<head>
<title> PHD Data Retrieval Using ASP</title>
</head>
<body BGCOLOR="LIGHTBLUE">
<font COLOR="BLUE" SIZE="3">
<p>Gets tag TIC21941.PV from host PC##.<br>
The time is <% = TIME %>. </font></p>
<p><%
Response.Write "<BR>Starting...<BR>"
Dim PHDOBJECT
Dim PHDTAGS
Set PHDOBJECT = CreateObject ("VisualPHD.Data")
PHDOBJECT.HOSTNAME = "PC##"
PHDOBJECT.LOGIN "TOTALPLANT","TOTALPLANT"
PHDOBJECT.TAGS.Add "TIC21941.PV"
PHDOBJECT.SAMPLEFREQUENCY = 60
PHDOBJECT.SAMPLEMETHOD = "SNAPSHOT"
PHDOBJECT.STARTTIME = "NOW"
PHDOBJECT.ENDTIME = "NOW"
PHDOBJECT.FETCH
RESPONSE.WRITE "Fetched data.<BR>"
RESPONSE.WRITE "Fetched " & PHDOBJECT.ROWCOUNT & " row(s).<BR>"
Set PHDTAGS = PHDOBJECT.TAGS("TIC21941.PV")
RESPONSE.WRITE "<BR>TIMESTAMP=" & PHDTAGS.TIMESTAMP
RESPONSE.WRITE "<BR>VALUE=" & PHDTAGS.VALUE
RESPONSE.WRITE "<BR>CONFIDENCE=" & PHDTAGS.CONFIDENCE
RESPONSE.WRITE "<BR><BR>Complete.<BR><BR>"
Set PHDOBJECT = NOTHING
%> <img SRC="COOLING%20TOWER.JPG" WIDTH="120" HEIGHT="120"> </p>
</body>
</html>
```

Designate ASP scripting with <% at the beginning and %> at the end.

Use the 'Response' ASP object to write to the web page.

VisualPHD must be on the web server for CreateObject to work.

If the program does not call the LOGIN method, the current operating system account is used to log on to the PHD server.

Exercise 1 - Thin Client

Overview

In this exercise, you will create a simple ASP script to access the current value of a PHD tag.

Instructions

1. On the web server machine ACPHD1, copy two files
   `SimpleASP1.asp` and `CoolingTower.jpg`

   from `C:InetPub/PC13` to `C:InetPub/PC##`

   Where: ## is your PC number.
   (`PC##` is your virtual directory, created by your instructor to segregate your files from other students' files. `PC13` is your instructor's virtual directory.)

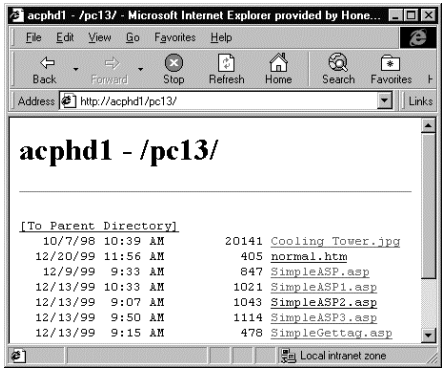
2. Rename the file `##SimpleASP1.asp` (where ## is your PC number).

3. To view the script, open the file in Front Page.

4. Using Front Page, modify the script to specify the name of your PHD server (PC##) instead of ACPHD1. Save the change.

5. To view your web page, open Internet Explorer and enter the URL for your virtual directory: `http://acphd1/pc##`

6. Select your file `##SimpleASP1.asp`.



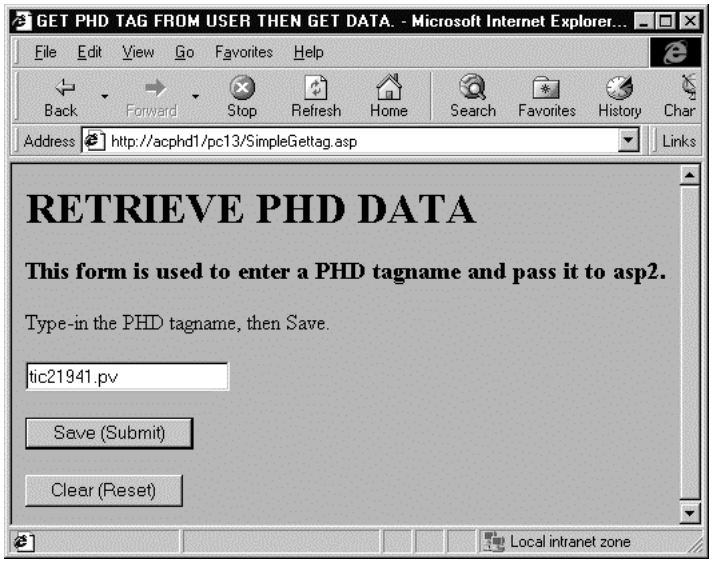
End of Exercise 1

Exercise 2 - Thin Client

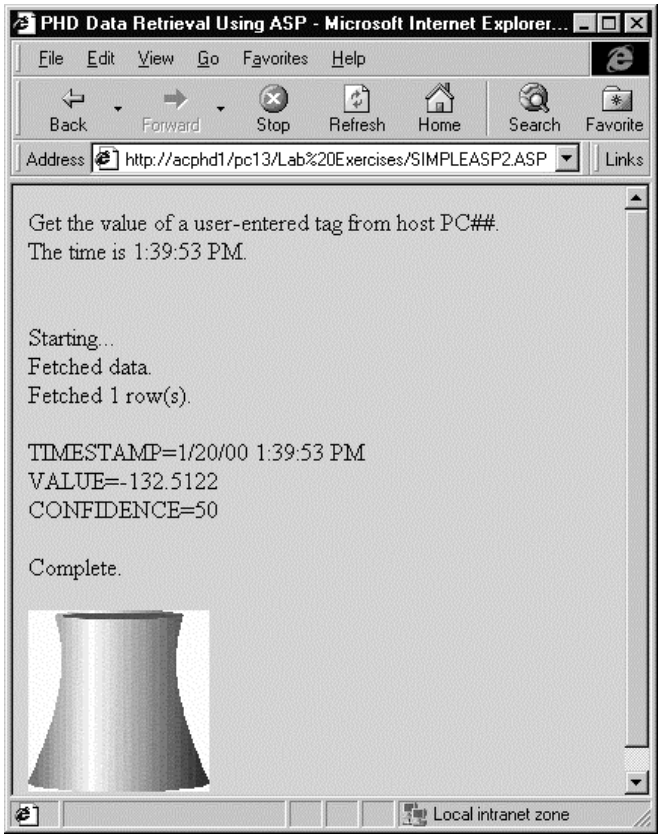
Overview

In this exercise, you will create two ASP scripts to enable the user to enter a PHD tagname.

##SimpleGetTag.asp



##SimpleASP2.asp



Exercise 2 - Thin Client

Instructions
Perform the following steps:

1. Using Notepad, modify your previous script (##SimpleASP1.asp) as shown in **bold** at the right.

2. Save the file as ##SimpleASP2.asp.

```
<html>
<head>
<title> PHD Data Retrieval Using ASP</title>
</head>
<body BGCOLOR="LIGHTBLUE">
<font COLOR="BLUE" SIZE="3">
<p>Get the value of a user-entered tag from a named host.<br>
The time is <% = TIME %>. </font></p>
<p><%
RESPONSE.WRITE "<BR>Starting...<BR>"
Dim PHDOBJECT
Dim PHDTAGS
Dim TAG
TAG = REQUEST.FORM("TAGNAME")
Set PHDOBJECT = CreateObject ("VisualPHD.Data")
PHDOBJECT.HOSTNAME = ″PC##"
PHDOBJECT.TAGS.Add TAG
PHDOBJECT.SAMPLEFREQUENCY = 60
PHDOBJECT.SAMPLEMETHOD = "SNAPSHOT"
PHDOBJECT.STARTTIME = "NOW"
PHDOBJECT.ENDTIME = "NOW"
PHDOBJECT.FETCH
RESPONSE.WRITE "Fetched data.<BR>"
RESPONSE.WRITE "Fetched " & PHDOBJECT.RowCount & " row(s).<BR>"
Set PHDTAGS = PHDOBJECT.TAGS(TAG)
RESPONSE.WRITE "<BR>TIMESTAMP=" & PHDTAGS.TIMESTAMP
RESPONSE.WRITE "<BR>VALUE=" & PHDTAGS.VALUE
RESPONSE.WRITE "<BR>CONFIDENCE=" & PHDTAGS.CONFIDENCE
RESPONSE.WRITE "<BR><BR>Complete.<BR><BR>"
Set PHDOBJECT = NOTHING
%> <img SRC="COOLING%20TOWER.JPG" WIDTH="120" HEIGHT="120"> </p>
</body>
</html>
```

Exercise 2 - Thin Client, *continued*

Instructions, *continued*

3. Create or copy the script shown below and save it as file ##SimpleGetTag.asp

```
<html>
<head>
<title>GET PHD TAG FROM USER THEN GET DATA.</title>
</head>
<body BGCOLOR="LITEGREEN">
<h1>RETRIEVE PHD DATA </h1>
<h3>This form is used to enter a PHD tagname and pass it to asp2.</h3>
<form ACTION="##SIMPLEASP2.ASP" METHOD="POST">
  <p>Type-in the PHD tagname, then Save.</p>
  <p><input TYPE="TEXT" NAME="TAGNAME" size="20"> </p>
  <p><input TYPE="SUBMIT" VALUE="Save (Submit)"></p>
  <p><input TYPE="RESET" VALUE="Clear (Reset)"></p>
</form>
</body>
</html>
```

Specify the name of your previous script.

4. Copy both asp files (##SimpleASP2.asp and ##SimpleGetTag.asp) to your assigned directory on the web server machine ACPHD1.
5. In Internet Explorer, enter the URL for your virtual directory: http://acphd1/pc##.
6. Select your file ##SimpleGetTag.asp and enter a tagname.

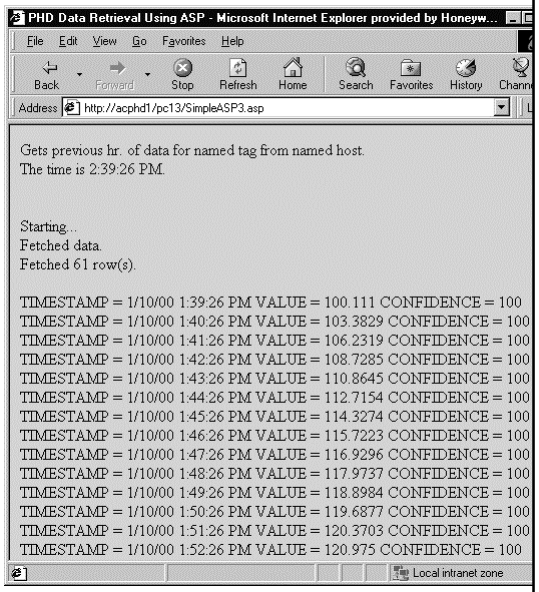
End of Exercise 2

Exercise 3 - Thin

Overview

Create an ASP script to access the previous hour of data for a tag.

1. Modify your first script (##SimpleASP1.asp) as shown in **bold** at the right.
2. Save the file as ##SimpleASP3.asp.




```
<html>
<head>
<title> PHD Data Retrieval Using ASP</title>
</head>
<body bgcolor="#00FFFF">
<p><font color="#0000FF" size="3">
<p>Gets previous hr. of data for TIC21941.PV from host PC##.<br>
The time is <% = TIME %>. </font></p>
<p><%
RESPONSE.WRITE "<BR>Starting...<BR>"
Dim PHDOBJECT
Dim PHDTAGS
Dim I
Set PHDOBJECT = CreateObject ("VisualPHD.Data")
PHDOBJECT.HOSTNAME = "PC##"
PHDOBJECT.LOGIN "TOTALPLANT","TOTALPLANT"
PHDOBJECT.TAGS.Add "TIC21941.PV"
PHDOBJECT.SAMPLEFREQUENCY = 60
PHDOBJECT.SAMPLEMETHOD = "SNAPSHOT"
PHDOBJECT.STARTTIME = "NOW-1H"
PHDOBJECT.ENDTIME = "NOW"
PHDOBJECT.FETCH
RESPONSE.WRITE "Fetched data.<BR>"
RESPONSE.WRITE "Fetched " & PHDOBJECT.ROWCOUNT & " row(s).<BR>"
Set PHDTAGS = PHDOBJECT.TAGS("TIC21941.PV")
FOR I = 1 TO 60
RESPONSE.WRITE "<BR>TIMESTAMP=" & PHDTAGS.TIMESTAMP
RESPONSE.WRITE "<BR>VALUE=" & PHDTAGS.VALUE
RESPONSE.WRITE "<BR>CONFIDENCE=" & PHDTAGS.CONFIDENCE
PHDOBJECT.MOVENEXT
NEXT
RESPONSE.WRITE "<BR><BR>Complete.<BR><BR>"
Set PHDOBJECT = NOTHING
%> <img src="COOLING%20TOWER.JPG" width="120" height="120"> </p>
</body>
</html>
```

Exercise 3 - Thin Client, *continued*

3. Copy the asp file ##`SimpleASP3.asp` to your assigned directory on the web server machine ACPHD1.
4. In Internet Explorer, enter the URL for your virtual directory: `http://acphd1/pc##`.
5. Select your file ##`SimpleASP3.asp`.

End of Exercise 3

Software Requirements

Without ASP

– **Client**

- **MS Windows 95/98 or NT 4.0**
- **MS Internet Explorer 3.02 or higher**
- **Editor:**
  MS Visual Basic 5.0 or 6.0 Development Edition
- **Visual PHD**
- **32 Bit ODBC Driver**

– **Server**

- **MS Internet Information Server 3.0 or higher**

With ASP

– **Client**

- **MS Windows 95/98 or NT 4.0**
- **MS Internet Explorer 4.0 or higher**
- **Editors:**
  MS FrontPage® 98 or 2000
  MS Visual Basic 6.0 Scripting Edition (Visual InterDev™)
  Macromedia DreamWeaver 2.0
  Notepad

– **Server**

- **MS Internet Information Server 4.0 or higher**
- **(see next page)**

Client machines need an ODBC connection to Oracle.

If users need to update the properties of the Visual PHD ActiveX controls through the browser, a "TotalPlant32" data source must be configured.

Software Requirements, *continued*

The following software should be installed on your web server:

1. OLE DB and ADO support - So ASP pages can access the Uniformance Oracle database
2. Oracle SQL*Net - Configured to provide access to the Uniformance Oracle database.
3. Visual PHD - The Visual PHD component of the Uniformance Desktop is essential.

Background

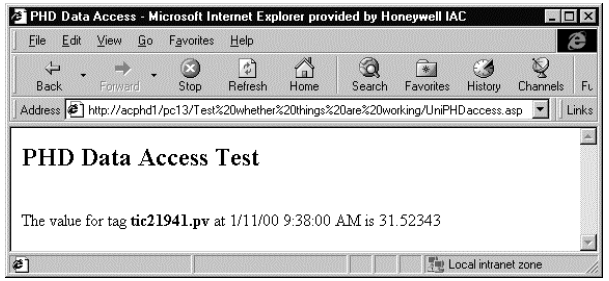
OLE DB and ADO support
OLE DB provides the database access, and ADO (Active Data Objects) is a programming interface on top of OLE DB. The best way to access the Uniformance Oracle database from web pages is by using ADO to connect to the Oracle database through OLE DB. The following software components are necessary and sufficient to provide the OLE DB and ADO support.

- **Database Access component** - An optional installable component of the Microsoft IIS web server product. Needed to provide the ADO interface to OLE DB database from web pages.
- **Microsoft Data Access SDK 2.0 or equivalent** - Needed to get the Oracle OLE DB provider and to update the Database Access component to the latest version.

Scripts to Test Installation

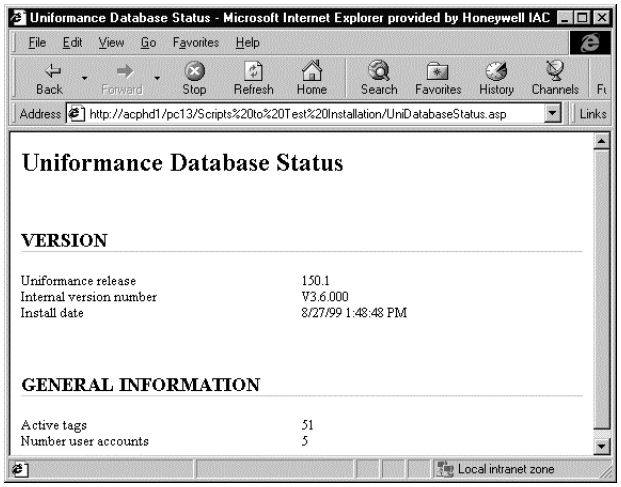
Available on your instructor's machine are files you can use to test an installation. You would copy all three files to a directory on the web server.

`UniPHDaccess.asp`



Fetches the current value of a single PHD tag. You will need to change line 9 to specify the TCP/IP host name of your PHD server, and line 12 to specify a tag to retrieve. Test it by invoking this page with your browser. If the Uniformance desktop software has been installed correctly on your web server, then you should see the current value for your selected tag.

`UniDatabaseStatus.asp`



Fetches some information from the Oracle database. To connect to the Microsoft Oracle provider, the script sets the provider argument of the ConnectionString property to MSDAORA. You may need to change line 11, depending on how SQL*Net is configured. If SQL*Net and the Database Access component have been installed correctly on your web server, then you should see a report showing the current Uniformance release and the number of active tags in your database.

`UniStyle.css` is an optional style sheet.

ActiveX Controls Vs OLE Server

- The Visual PHD OLE server exposes a number of properties and methods in standard OLE fashion. It does not have a user interface. The only way to talk to it is with a programming language such as VB.
- The ActiveX controls have a user interface and configuration dialogs, and are designed for drag and drop use.
- ASP server-side scripts get PHD data programmatically, then handle all data presentation in HTML.
- If you want to handle data presentation through a programming language, you do not need the ActiveX controls. In fact, it may be easier to fetch data programmatically by talking to the OLE Server than to fetch data programmatically by talking to the ActiveX controls.

Functionality	ActiveX controls	OLE Server
Supports programming to access data	yes (but limited)	yes
Supports drag/drop use so that programming is optional	yes	no
Handles formatting and presenting results	yes	no (this can be good if fine control is desired)
Can run on web client, accessing data from client machine	yes	yes
Can run on web server, accessing data from server machine	no	yes
Performance implications when requesting multiple values	slow - each is treated as an individual request	faster - multiple requests can be consolidated
User interface (dialogs)	yes	no

Hands-on Exercise - Fat Client

Overview

In this exercise, you will create a VB application containing Visual PHD ActiveX controls, then convert the application to an ActiveX document. You will create an Internet Download Setup for the ActiveX document using the VB Package and Deployment Wizard.

Shown below is an overview of the steps you will perform for this exercise.

Part I: Create an ActiveX Document in Visual Basic

1. **Open a new ActiveX Document**
2. **Add Visual PHD Controls to the VB General Toolbar**
3. **Place the Visual PHD Controls on the ActiveX Document**
4. **Add Graphics to Form**
5. **Save and Compile the Project**

Part 2: Place the ActiveX Document on the Intranet

6. **Create an Internet Download Setup**
7. **Copy the Compiled Project Files to the Internet/Intranet Directory**
8. **View the ActiveX Document**

Hands-on Exercise - Part I - Create ActiveX Document

Instructions

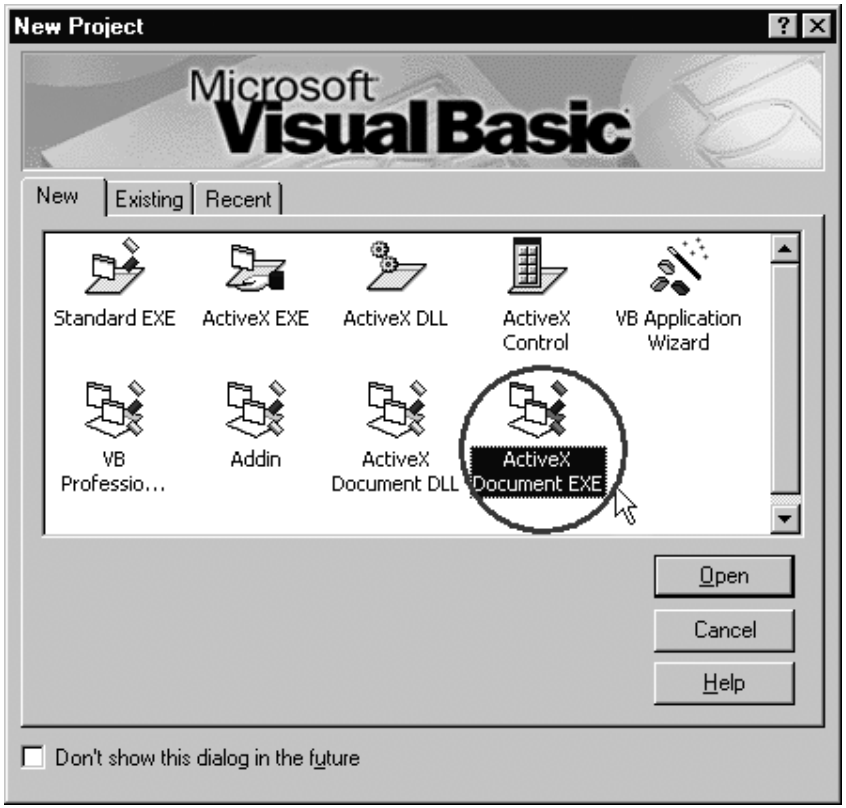
1.1. Open a new ActiveX Document

In this task, you will create an **ActiveX .exe**:

1.1.1. Open Visual Basic 6.0.

1.1.2. Choose **ActiveX Document EXE** from the **New** tab in the **New Project** dialog.

1.1.3. Select the **Open** button to open the new document.



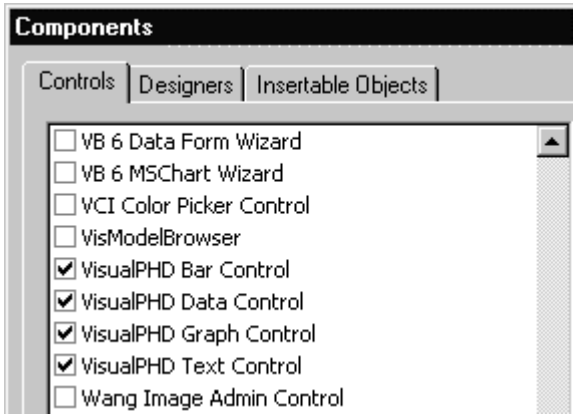
Hands-on Exercise - Part I - Create ActiveX Document

1.2. Add Visual PHD Controls to the VB General Toolbar

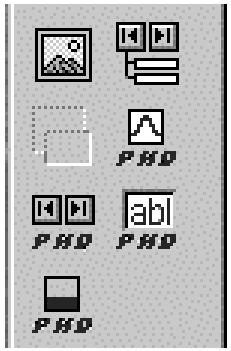
1.2.1. Right-click on the **General** Toolbar.

1.2.2. Select the **Components…** menu option.

1.2.3. Scroll to the Visual PHD Controls and select all of the Visual PHD Controls.



1.2.4. Select the OK button to apply your changes and exit the window.
Result: *The Visual PHD controls will now appear at the bottom of your toolbar.*



Each time you create a new ActiveX Document, the Visual PHD controls will have to be added to the toolbar.

Hands-on Exercise - Part I - Create ActiveX Document

1.3. Place the Visual PHD Controls on the ActiveX Document

Now that the controls have been placed on the toolbar, we are ready to place the controls on the document.

In this task, you will add and configure the Data and Text Controls:

1.3.1. Expand the **UserDocuments** folder in the **Project** window.

1.3.2. Double-click the **UserDocument1** form to open the form.

Hands-on Exercise - Part I - Create ActiveX Document

1.3.1. Add Data Control

The PHD Data control retrieves data from PHD. The other controls (text, bar, and graph) rely on the PHD Data control for their data.

1.3.1.1. Drag the lower right corner of the form to enlarge it to at least 6” x 6”.

1.3.1.2. Click on the **PHD Data Control** and add it to the form by placing the crosshair cursor at the upper left corner where you want the control to appear and click and drag a rectangle.

1.3.1.3. Right-click on the Data Control and select the **Properties** menu option.

1.3.1.4. Type the name of your PHD Server machine as the **PHD Hostname**. Select OK.

Hands-on Exercise - Part I - Create ActiveX Document

1.3.2. Add Text Control

The Visual PHD Text control displays PHD values.

1.3.2.1. Click on the **PHD Text Control** and add it to the form by placing the crosshair cursor at the upper left corner where you want the control to appear and click and drag a rectangle of at least 1".

1.3.2.2. Right-click on the Text Control and select the **Properties** menu option.

1.3.2.3. Select **PHDCtrl1** as the Data Source name assigned to the Data Control.

1.3.2.4. For the **Tagname** enter **TIC21###.PV**, where ### is your assigned number.

1.3.2.5. Select the OK button to apply your changes and exit the window.
Result: *The Text Control has been added and configured.*

Hands-on Exercise - Part I - Create ActiveX Document

1.4. Save and Compile the Project

1.4.1. Select the **Save Project** menu option from the **File** menu.

1.4.2 Browse to the **D:\Program Files\Honeywell TPI\VisualPHD** directory on your PC.

1.4.3. **UserDocument1.dob** appears as the **File name**. Select the **Save** button.

1.4.4. **Project1.vbp** appears as the **File name**. Select the **Save** button.

1.4.5. Compile the project by selecting the **Make Project1.exe …** menu option from the **File** menu.
Navigate to **D:\Program Files\Honeywell TPI\VisualPHD** .

1.4.6. **Project1.exe** appears as the filename in the **Make Project** window. Select the OK button

Result: *The project has now been saved and compiled.*

1.4.7. Exit Visual Basic.

2.1. Create an Internet Download Setup

Before the ActiveX document is placed on the Intranet, you must create an Internet Download Setup. The .CAB file created by the setup will automatically copy your ActiveX Document from the Internet Server to your PC client when opened in a browser.

2.1.1. Start the **Package and Deployment Wizard** from the **Microsoft Visual Basic 6.0 Tools** folder or program group.

2.1.2. Select the **Browse...** Button on the Package and Deployment dialog and find your project (it should be in **D:\Program Files\Honeywell TPI\VisualPHD**).
Your project file will have the extension .VBP

Select the **Open** Button on the Open Project dialog.

2.1.3. Select the **Package** button on the Package and Deployment Wizard dialog.

2.1.4. Select **Internet Package** as the package type.

2.1. Create an Internet Download Setup, continued

2.1.5. Select the **Next** button.

2.1.6. Select the **Next** button to accept the default location.

2.1.7. Select the **YES** button to create the folder named ‘Package’.

2.1.8. Select the **NO** button to not include the Remote Automation server files.

2.1.9. Select the **OK** button to proceed without the missing file.

(AutPrx32.dll is not used in this web page)

2.1.10. Select the **OK** button on the Missing Dependency Information dialog.

NOTE:
The ”Missing Dependency Information" window displays the Visual PHD controls used in the document. If these files are selected, client access to the web page will cause Internet Explorer to download and register the Visual PHD files. Unfortunately, this function does not register the Visual PHD files correctly and does not place the ocx's in the correct directories. Visual PHD, for this reason, must be installed on the client's computer using the Visual PHD installation program.

2.1. Create an Internet Download Setup, continued

2.1.11. Select the **OK** button to proceed without the missing file. (AutMgr32.exe is not used in this web page)

2.1.12 Select the **OK** button to proceed without the missing file. (RACMgr32.exe is not used in this web page)

2.1.13. Uncheck the PHD ocx's on the Included Files dialog.

2.1.14. Select the **Next** button to continue.

2.1.15. In the File Source dialog, select the file **VB6 Runtime and OLE automation,** then select the file source '**Include in this cab'**.

2.1.16. Select the **Next** button to continue.

2.1.17 In the Safety Settings dialog, select **YES** for 'Safe for Scripting and **YES** for 'Safe for Initialization'.

2.1.18. Select the **Next** button to continue.

2.1.19. Select **Finish** to create the cab file, then **Close**.

2.2. Copy the Compiled Project Files to the Intranet Directory

Now you must copy the prepared ActiveX document to the Internet Information Server machine (ex. ACPHD1).

2.2.1. Select the **Deploy** button.

2.2.2. Select the **Next** button to accept the default Package to Deploy.

2.2.3. Select **Folder** in the Deployment Method dialog.

2.2.4. Select the **Next** button to continue.

2.2.5. In the Folder dialog, select **Network**, then navigate to the **C:\InetPub/PC##** directory on the Internet Information Server machine (ACPHD1), where ## is your PC number (1-12).

2.2.6. Select the **Next** button to continue.

2.2.7. Select **Finish** to create your Internet download files.

2.2.8. Select the **Close** button and **Close** again**.**

Hands-on Exercise - Part II - Place ActiveX Document on Intranet

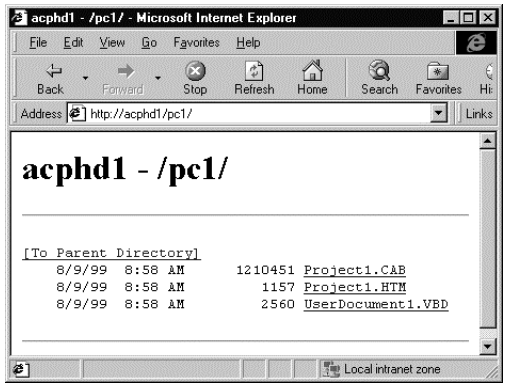
2.3. View the ActiveX Document

2.3.1. Open Internet Explorer.

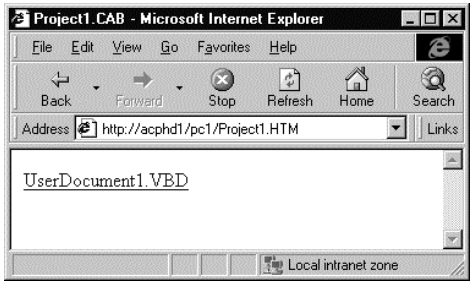
2.3.2. In the **Address** text box, enter: **`http://acphd1/pc##`**

2.3.3. Select your **.htm** file.

The .htm file contains a link to your VB document.



2.3. View the ActiveX Document, continued



2.3.4. Select the link to **UserDocument1.VBD.**

Result: *The ActiveX Document and all associated files are downloaded to your local PC.*

2.3.5. Select the exclamation point button on the Data control to begin data retrieval.

- *The Data control will by default return data for the last day.*

2.3.6. Right-click on the PHD controls and experiment by changing their **Properties**.

- *When the Data Control is in the PLAY mode, you cannot change control properties.*

End of Exercise

Terminology

ActiveX Control
An ActiveX control is a software component that performs common tasks in standard ways. ActiveX controls are an architecture for developing programmable software components to use in a variety of different containers, including Visual Basic and Excel.

CAB (Cabinet) File
A **.cab** file contains compressed files that are necessary for distributing and installing ActiveX components/applications.

Visual PHD
Visual PHD provides retrieval of PHD data from development tools such as Visual Basic. Visual PHD consists of an OLE Automation Server to retrieve data from and put data to PHD, and a collection of ActiveX (OCX) controls to visually access data from PHD. The Visual PHD ActiveX controls call the PHD OLE Server to get PHD data, and call Oracle directly (using SQL*Net and, possibly, ODBC calls) to get Oracle data.

ODBC
Microsoft's standard for accessing databases. It is a C-language API that any application can call, plus a set of APIs that you can plug-into the standard if you have your own database. MS provides the standard and with Windows they provide supporting tools, such as tools to supply the alias names.

OLE DB
OLE DB is the next generation of standards for accessing databases. It imposes less about the structure of the database. Uniformance has written an OLE DB provider for PHD--the code that exposes the OLE DB interface (such as apps calls and PHD function calls). An OLE DB provider is any software component that exposes an OLE DB interface.

ADO
ADO is the syntax used to access a database. It is the programming language Microsoft provides to OLE DB providers. To use OLE DB directly, you have to be a C++ programmer. ADO is available in scripting languages and VB at a slightly higher layer on top of it..

Microsoft supplies an Oracle provider for OLE DB. It allows ADO to access Oracle databases. Microsoft also makes an ODBC provider (so you can use ADO programming to access an ODBC datasource), a SQL server provider, and the jet engine provider for a Microsoft Access database.

Honeywell

Helping You Manage Your World

www.iac.honeywell.com