OBJECTSTORE

ACTIVE TOOLKIT TUTORIAL

RELEASE 6.0

May 1999

ObjectStore Active Toolkit Tutorial

ObjectStore Active Toolkit Release 6.0, May 1999

Object Design, Inc.
Twenty Five Mall Road
Burlington, MA 01803-4194

Contents

Preface

Purpose

The *ObjectStore Active Toolkit Tutorial* demonstrates how to use the Active Toolkit (ATK) ActiveX and OLE DB programming interface to ObjectStore.

Audience

This tutorial is for experienced Visual Basic, VBScript, Java, JavaScript, or C++ developers who are developing applications that run under Windows NT or Windows 98 and use an ActiveX interface to access objects stored in an ObjectStore database. It assumes some familiarity with ObjectStore Inspector Release 6.0.

Scope

This book supports Release 6.0 of the ATK interface to ObjectStore Release 6.0. Information in this book assumes that ATK is installed and configured.

How This Tutorial Is Organized

The tutorial has six modules:

Chapter	*Contents*
Chapter 1, A Visual Basic Application with ATK ActiveX Server, on page 1	Create a Visual Basic application that uses the ATK ActiveX server to retrieve tabular information from an ObjectStore database.
Chapter 2, Accessing ATK ActiveX Server from ASP Applications, on page 21	Create Active Server Page applications that use the ATK ActiveX server to retrieve tabular information from an ObjectStore database. Allow the user to navigate from one type of data to another.

Chapter	*Contents*
Chapter 3, Accessing ATK OLE DB Provider from Active Server Page Applications, on page 47	Create Active Server Page applications that use the ATK OLE DB data source to retrieve tabular information from an ObjectStore database.
Chapter 4, Creating an ODBC Data Source Using ATK, on page 69	Access the ObjectStore Active Toolkit OLE DB provider from an ODBC-compliant reporting tool.
Chapter 5, Using Crystal Reports with ATK, on page 79	Create a data view that you can use with any reporting tool. Then, use Crystal Reports to create a report.
Chapter 6, Using ATK ActiveX Server from DCOM, on page 91	Configure DCOM and test its remote connection between an ATK ActiveX grid control and an ATK ActiveX server.

Sample Data

This tutorial refers to sample applications and demonstration databases. If you installed ATK using the installation program defaults, you can find them in these directories:

Component	*Location*
ATK	**C:\odi\ATK6.0**
Demonstration Database	**C:\odi\ATK6.0\Examples\demodbs**

All examples in this tutorial refer to these locations.

Notation Conventions

This document uses the following conventions:

Convention	*Meaning*
Bold	Bold typeface indicates user input or code.
Comment	Comment highlights code comments.
Sans serif	Sans serif typeface indicates system output.

Convention	*Meaning*
Italic sans serif	Italic sans serif typeface indicates a variable for which you must supply a value. This most often appears in a syntax line or table.
Italic serif	In text, italic serif typeface indicates the first use of an important term.
[ ]	Brackets enclose optional arguments.
{ *a* | *b* | *c* }	Braces enclose two or more items. You can specify only one of the enclosed items. Vertical bars represent OR separators. For example, you can specify *a* or *b* or *c*.
...	An ellipsis indicates missing code that is not pertinent to the current example. In syntax lines, it indicates that the previous item can be repeated.

Internet Sources of More Information

Object Design

Object Design's site on the World Wide Web is the source for company news, white papers, and information about product offerings and services. Point your browser to **http://www.objectdesign.com/** for more information.

Other ObjectStore products

In addition to ObjectStore, the industry's leading object database, Object Design offers a comprehensive set of rapid development and enterprise integration tools. For information about these and other Object Design products, point your browser to **http://www.objectdesign.com/products/products.html**.

Product support

Object Design's support organization provides a number of information resources and services. Their home page is at **http://support.objectdesign.com/**. From the support home page, click Tech Support Information to learn about support policies, product discussion groups, and the different ways Object Design can keep you informed about the latest release information — including the web, **ftp**, and email services.

Training

If you are in North America, for information about Object Design's educational offerings, call 781.674.5320, Monday through Friday from 8:30 AM to 5:30 PM Eastern Time. Outside these hours, call 800.706.2507.

If you are outside North America, call your Object Design sales representative.

Your Comments

Object Design welcomes your comments about ObjectStore documentation. Send your feedback to **support@objectdesign.com**. To expedite your message, begin the subject with **Doc:**. For example:

Subject: Doc: Incorrect message on page 76 of reference manual

You can also fax your comments to 781.674.5440.

Chapter 1
A Visual Basic Application with ATK ActiveX Server

Introduction

Using ATK ActiveX server, you can display tabular views from an ObjectStore database without coding a complex application. ATK provides an object model that you can use with any ActiveX-compliant development tool, such as Visual Basic. Using the classes and methods in the object model, you can easily inspect the database from the development environment that best meets your needs.

Software requirements

To complete the exercises in this chapter, you need these software resources:

Resource	*Where to Find It*
Database	**\odi\ATK6.0\Examples\demodbs\carsdemo.db**
Visual Basic projects	**\odi\ATK6.0\Examples\Tutorial1**

In this chapter

In this chapter, you create a Visual Basic application that uses the ATK ActiveX server to retrieve tabular information from an ObjectStore database. This chapter contains these exercises:

Exercise	*Description*
Fill a List Box with Data on page 2	Open an ObjectStore database and display data from a specific data view.
Fill a Grid Control with Data View Content on page 14	Display the contents of a data view in a grid control.

Fill a List Box with Data

Overview

The simple Visual Basic application you construct in this chapter opens the ObjectStore **carsdemo.db** database. It uses a list box to display the contents of a data view you define using ObjectStore Inspector.

Tip: In order for the ATK ActiveX server to retrieve data from a data view in the **carsdemo.db** database, the metaknowledge must contain the definition of the data view.

Process

To fill a list box with data, follow these steps:

1 Create a new data view to display in the Visual Basic application.

2 Customize the instance format.

3 Save the instance format as a grid template.

4 Save the new data view.

5 Create the Visual Basic ATK ActiveX Client.

6 Reference the ATK type library.

7 Write Visual Basic code to access the ATK ActiveX server.

8 Test the application.

9 Order the data by work order number.

10 Save the new data view.

11 Use the new data view.

12 Test the application.

Building the Application

1 Create a new data view to display in the Visual Basic application.

1 Start Inspector.

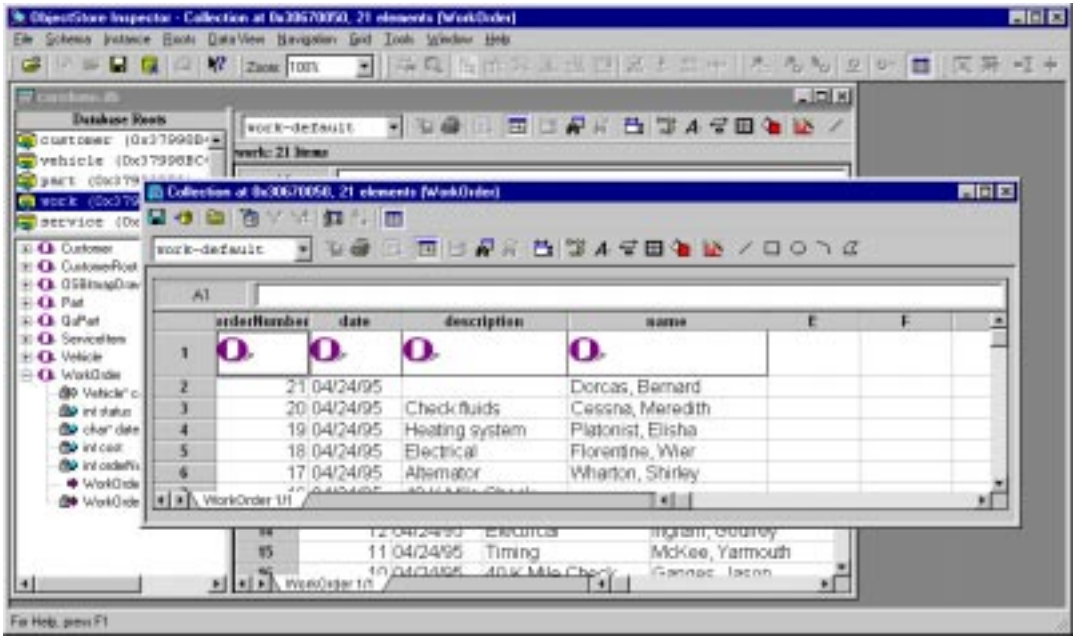
2 Open the ATK sample database **\odi\ATK6.0\Examples\DemoDBs\carsdemo.db**.

3 To create a data view based on the **work** root, double-click on the **work** root name in the **Database Roots** pane.

The instance pane is populated with the extent associated with the **work** root.

4 Select **Data View | Create** from the menu bar.

Inspector displays the new data view in an untitled Data View window.



2 Customize the instance format.

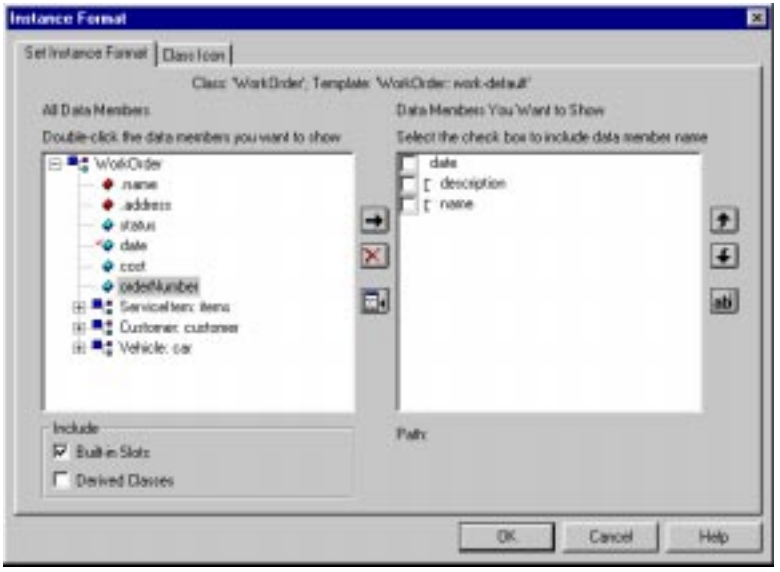
The *instance format* is a way to specify which data members are available to the application. You can save an instance format explicitly, as part of a *grid template*, or implicitly, as part of the data view.

1 Right-click anywhere in the data view grid.

2 Select **Set Format of Class** from the shortcut menu.

The Instance Format dialog box appears.

The left pane of the **Instance Format** sheet lists all data members in the data view; the right pane shows the members that have been included in this instance format.

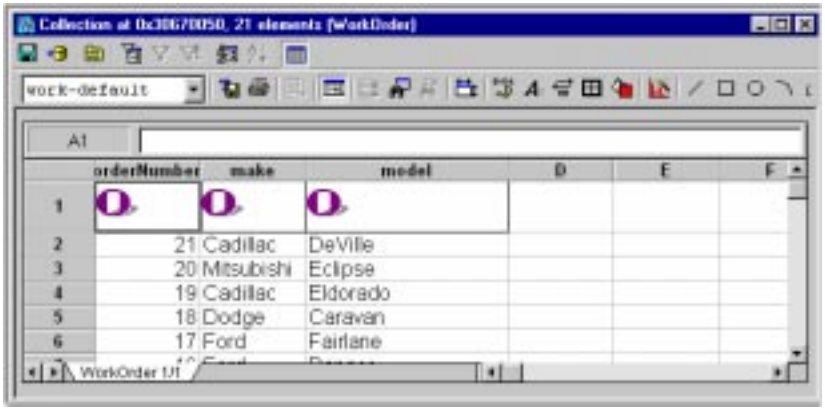


3 Include **orderNumber** in the instance list: select **orderNumber** from the **WorkOrder** class in the left pane, and click on the right arrow button.

4 Click **car** to display all members of that class that can be implicitly navigated, or reached by means of **car**. (For example, **make**, **model**, and **year** are data members that are related to **car**.)

5 Include the **make** and **model** in the instance format. This allows implicit navigation from **car** to the **make** and **model** of the related **Vehicle** instance.

The data view is refreshed based on these changes and now shows only the **make** and **model**.



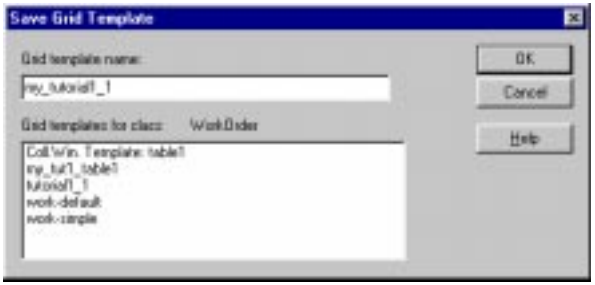
3 Save the instance format as a grid template.

As mentioned earlier, you can save instance format information explicitly, with a grid template, or implicitly, as part of the data view. Instance format information saved with a grid template can be used by ATK — in fact, you can have ATK select any instance format associated with a data view when the instance format is saved with a grid template.

1 Select **Grid | Template | Save As** from the menu bar.

The Save Grid Template dialog box appears.

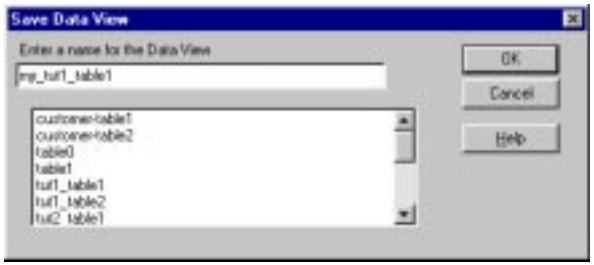
2 Name the new instance format **my_tutorial1_1**.



4 Save the new data view.

1 Select **File | Save All** from the menu bar.

2 Name the new data view **my_tut1_table1**.



5 Create the Visual Basic ATK ActiveX Client.

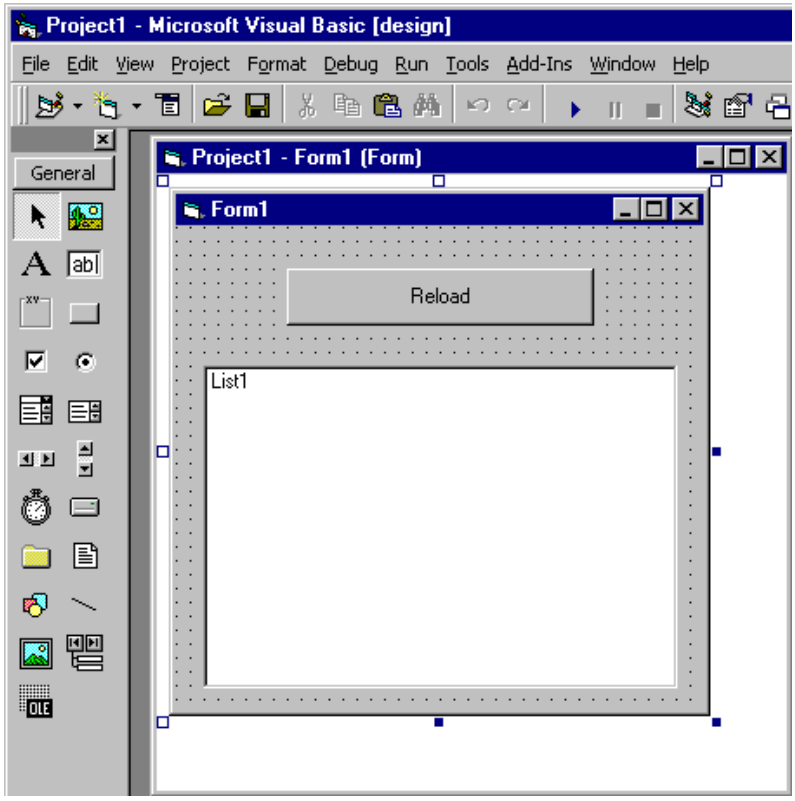
Now that you have created an instance format, create a Visual Basic ATK ActiveX client to display the data. Build a form with a list box that displays the items from **my_tut1_table1** and a button to load and refresh the data in the list box.

1 Start your Visual Basic development environment application, and create a new project.

Visual Basic displays an empty form.

2 In the form, create a new list box.

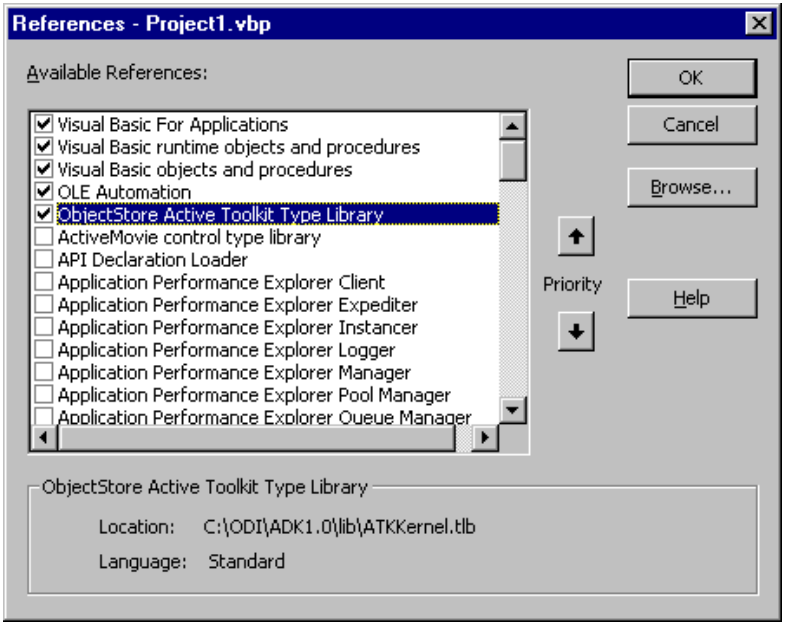
3 Create a button with the caption property **Reload**.



6 Reference the ATK type library.

1 In the Visual Basic environment, select **Project | References**.

2 Verify that **ObjectStore Active Toolkit Type Library** is checked in the **Available References** list.



If you cannot find the library, click on the **Browse** button and go to the **odi\ATK6.0\lib** directory. Select **ATKKernel.tlb**, the type library file. The **ObjectStore Active Toolkit Type Library** appears in the list.

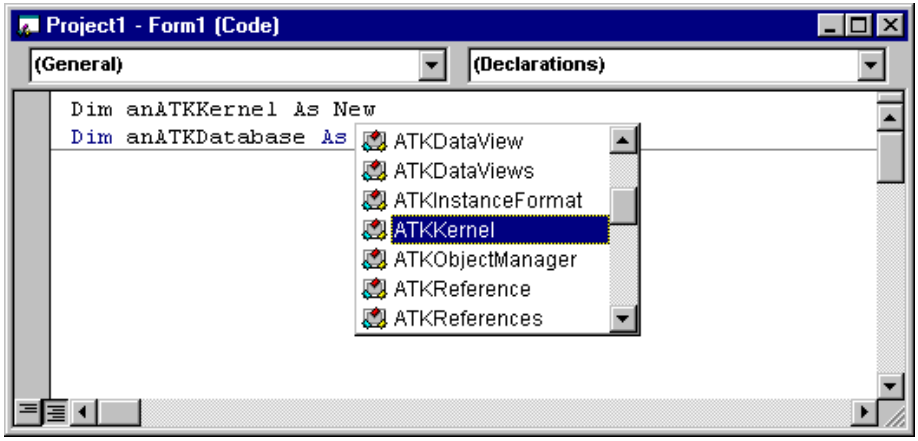
7 Write Visual Basic code to access the ATK ActiveX server.

1 In the **Declarations** block of the Visual Basic form, declare two global variables containing **ATKKernel** and **ATKDatabase** objects:

```
Dim anATKKernel As New ATKKernel
Dim anATKDatabase As ATKDatabase
```

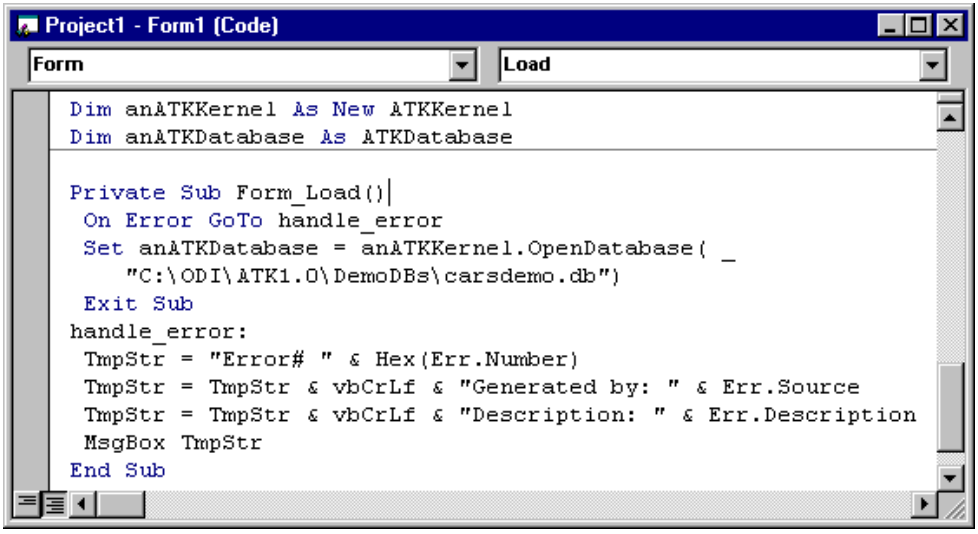
The variable **anATKKernel** is also instantiated to an **ATKKernel** object.

Tip: Because you referenced the ATK type library in your Visual Basic project, these ATK data types are available in the drop-down list box:

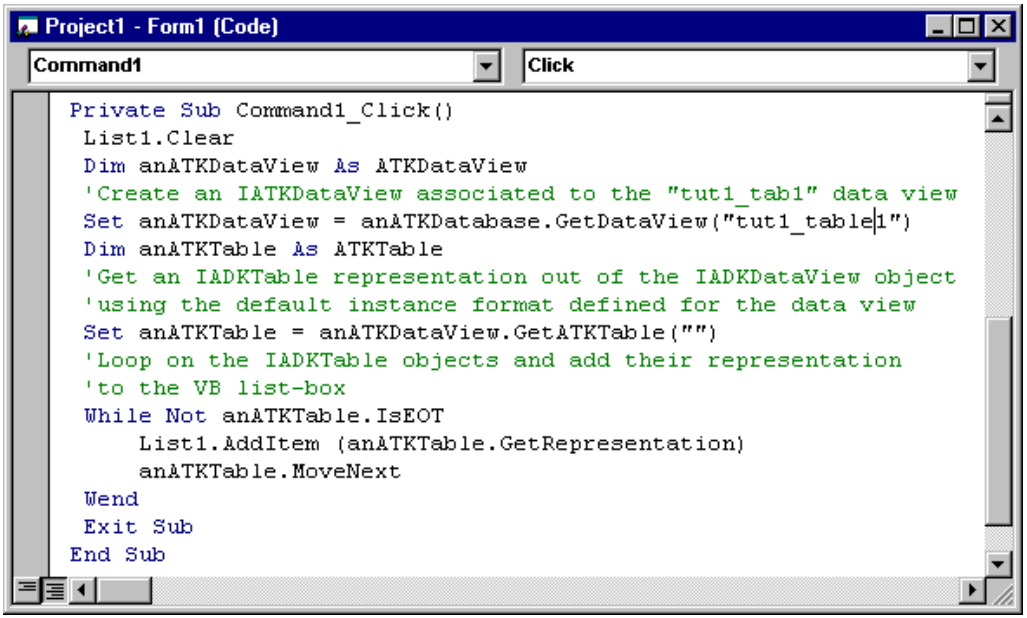


2 When the application starts, the form must be initialized with the appropriate data. Create a private subprocedure called **Form_Load**, using the **Form1** code example as a model, to open the **carsdemo.db** database.

3 The application must be prepared to handle possible errors. Enter the **handle_error** code to do this:



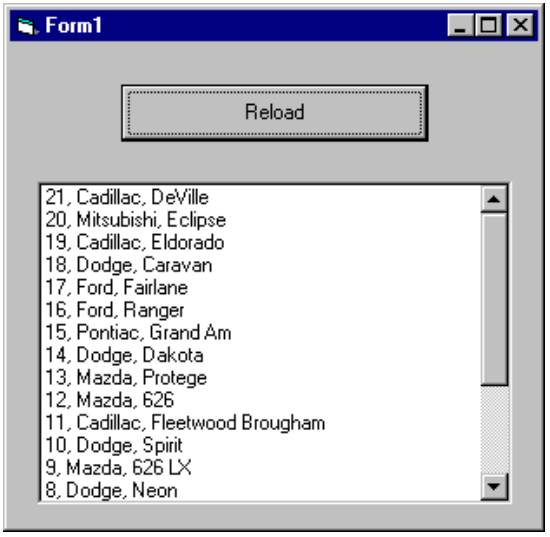
4 Create a private subprocedure called **Command1_Click**, using the code in the next screen as a model, to refresh the list box. When the user clicks the **Reload** button, the application must fill the **List1** list box with data. ATKTable::GetRepresentation "flattens" the table columns for display as a single string.



8 Test the application.

1 Run the Visual Basic application.

2 Click the **Reload** button. Data from **my_tut1_table1** appears in the list box.



9 Order the data by work order number.

Modify the index path in the **my_tut1_table1** data view. An *index path* is an ordered set of data members that users can navigate. Specify an index path that provides navigation from the **WorkOrder** to the **WorkOrder::orderNumber** data member.

1 Open the **my_tut1_table1** data view in Inspector.

2 Select **Data View | Define Order**.

3 Select **WorkOrder::orderNumber** as the index path:

ATK orders the data view accordingly.

10 Save the new data view.

1 Select **File | Save As**.

The Save Data View in Database dialog box appears.

2 Type the name **tut1_table2** and click the **OK** button.

11 Use the new data view.

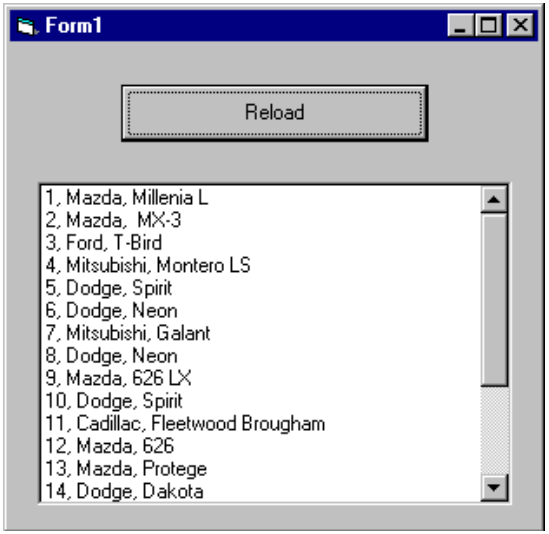
To make the Visual Basic application open the new **my_tut1_table2** data view, modify the command that retrieves the data view by changing the **Set anATKDataview** command. The change requires you to substitute **my_tut_table2** for the previous entry, **mytut1_table1**.

```
...

Set anATKDataView = anATKDatabase.GetDataView("my_tut1_
table2")

...
```

12 Test the application.

1 Run the Visual Basic application.

2 Click the **Reload** button. Data from **my_tut1_table2** appears in the list box, in ascending order by **orderNumber**.



Sample Code

The Visual Basic code for this example is available in **\odi\ATK6.0\Examples\Tutorial1\tut1_1.frm**.

Fill a Grid Control with Data View Content

Overview

This simple Visual Basic application displays the contents of a data view in a grid control.

Process

To fill a grid control with data view content, follow these steps:

1 Insert a grid control.

2 Customize the grid control.

3 Change the code that loads the form.

4 Modify the code that handles the button click.

5 Test the application.

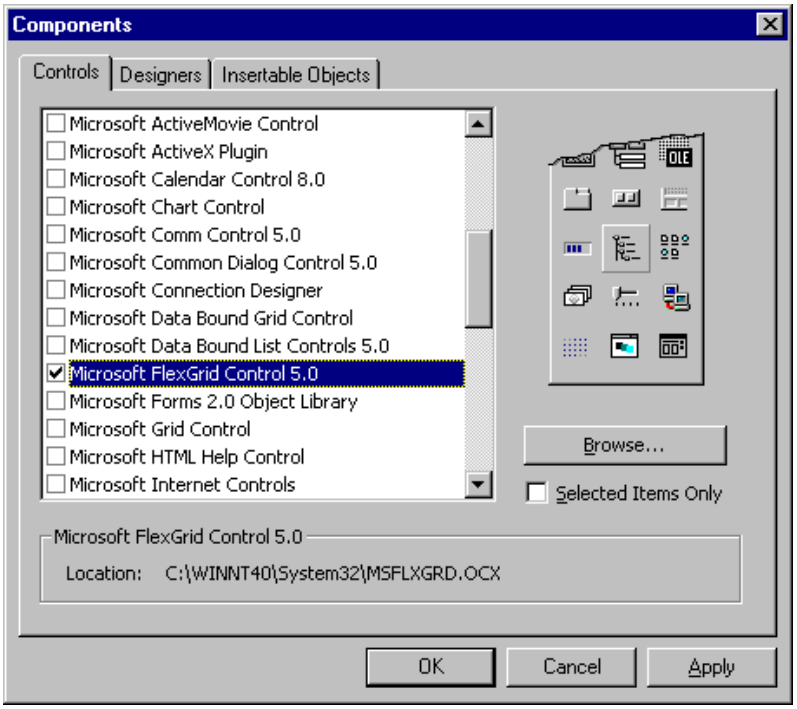
1 Insert a grid control.

Visual Basic 5.x contains a **MSFlexGrid** grid control that displays and operates on tabular data.

1 Delete the **List1** control from your form.

2 To insert an **MSFlexGrid** control, select **Project | Component**.

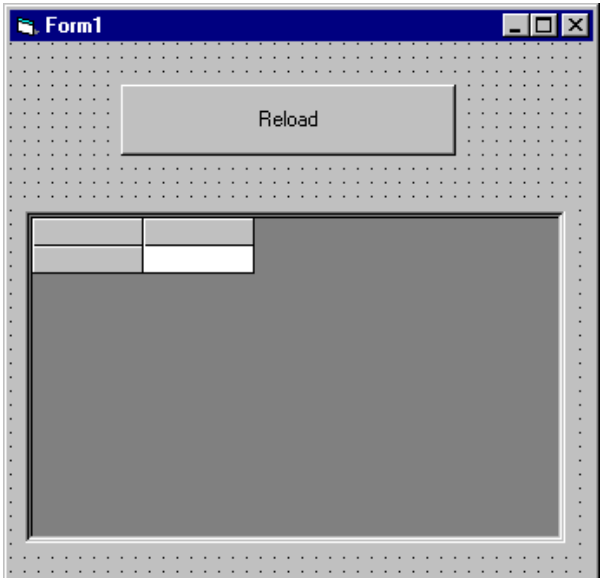
3 On the **Controls** sheet of the **Components** dialog box, check **Microsoft FlexGrid Control 5.***x*



The **MSFlexGrid** icon appears on the **Controls** toolbar: 

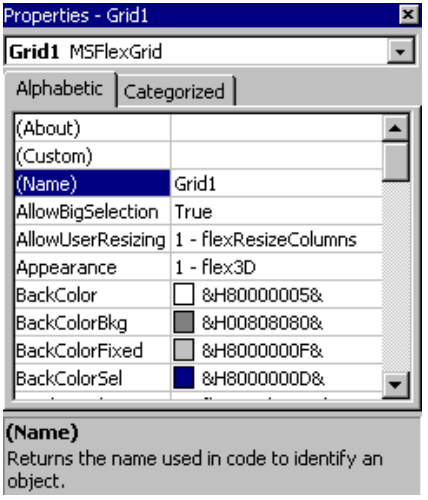
4 To create an **MSFlexGrid** control in your form, click on the **MSFlexGrid** icon.

The **Form1** dialog box appears



2 Customize the grid control.

1. Name the new grid control **Grid1**.
2. Set the **AllowUserResizing** property to **flexResizeColumns**.

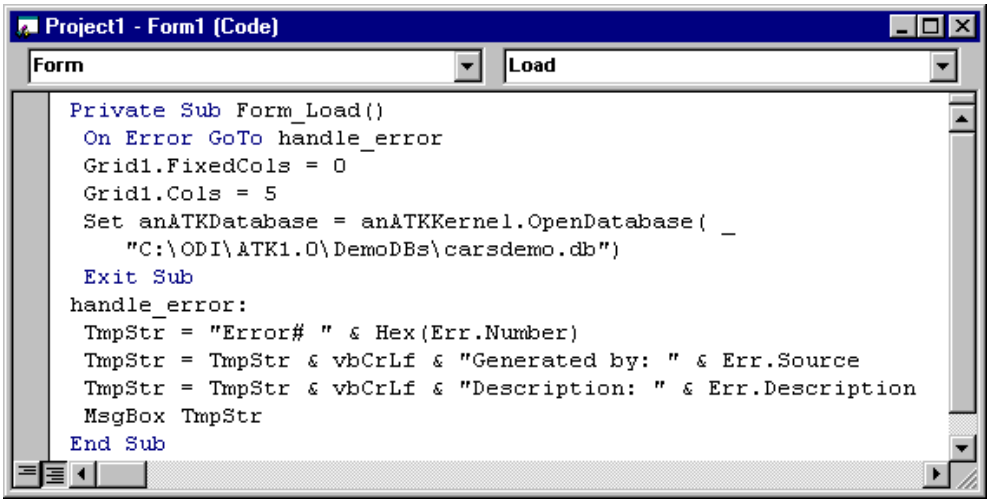


3 Change the code that loads the form.

1 Add these two lines to the **Form_Load** procedure:

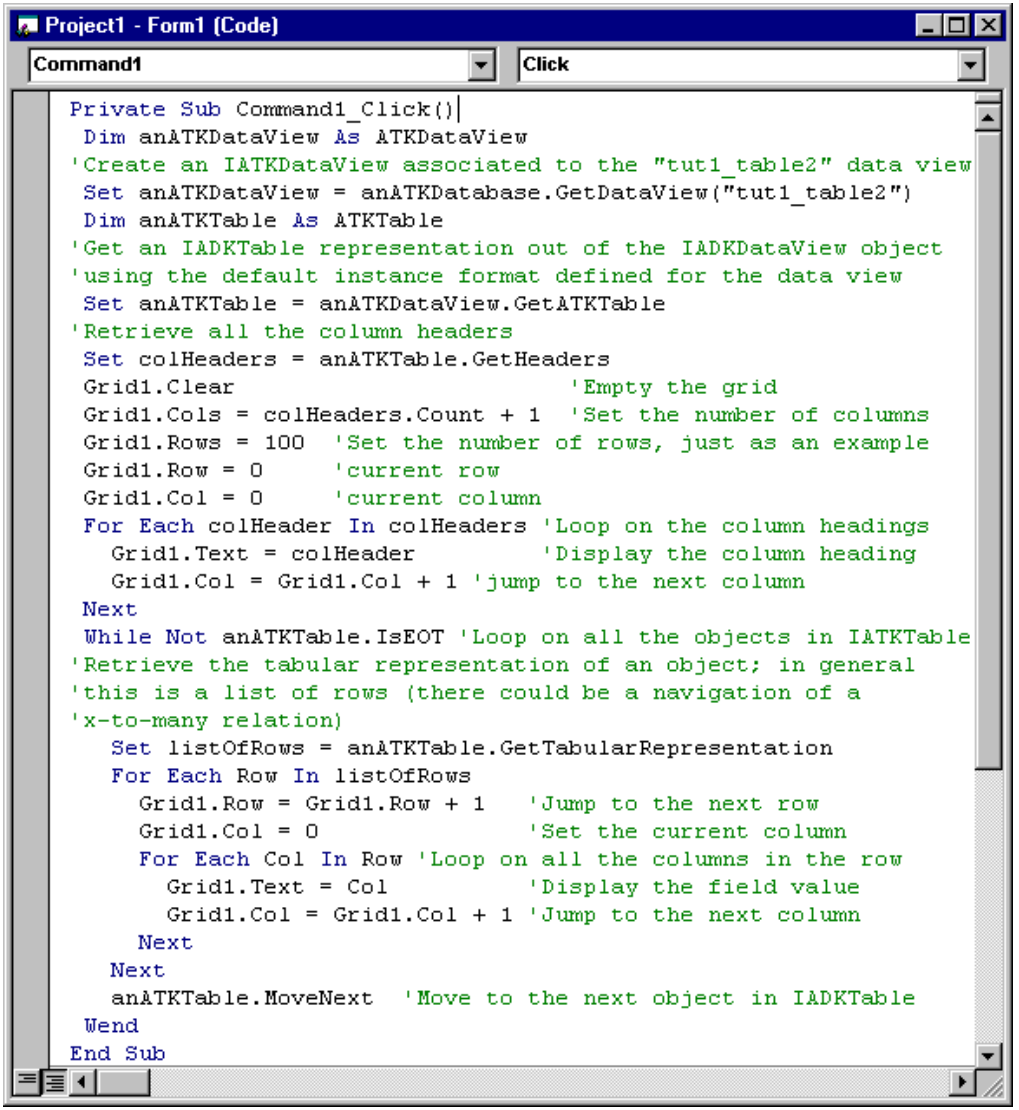
```
Grid1.FixedCols = 0
Grid1.Cols = 5
```

The value 5 is a reasonable default for the number of columns in the grid. The code now looks like this:



4 Modify the code that handles the button click.

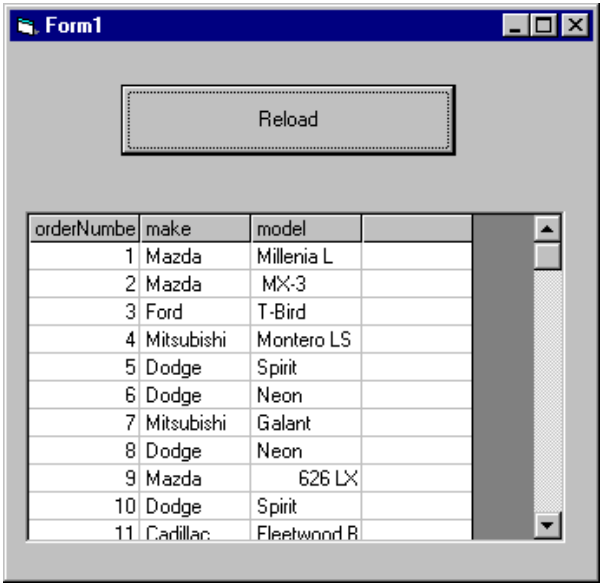
1 To retrieve the tabular representation of each object in the data view, and load data from each field into the proper grid cell, modify **Command1_Click**, as shown below, to refresh the grid.



5 Test the application.

1 Run the Visual Basic application.

2 Click the **Reload** button. Data from **my_tut1_table2** appears in the list box, in ascending order by **orderNumber**.



Sample Code

The Visual Basic code for this example is available in **\odi\ATK\Examples\Tutorial1\tut1_2.frm**.

Summary

In this chapter, you used ObjectStore Inspector to create a *data view* of the **carsdemo.db** database. You used the data view to define the data you wanted to query and a *grid template* to define the *instance format*, which controls the data member display.

Next, you created a Visual Basic application to open the ObjectStore database, query the data view you created, and display the returned data in a list box on a form.

Then, you replaced the list box with a grid control, modified the Visual Basic code to display the data in that grid control, and implemented a **Reload** button to refresh the data.

Although this chapter focuses on using ATK with Visual Basic, these examples are also portable to other development environments that support ActiveX, such as Microsoft J++ or Borland Delphi.

Chapter 2
Accessing ATK ActiveX Server from ASP Applications

Introduction

Active Server Page (ASP) allows you to write scripts that run on a server and provide standard HTML output. Although ASP is available only in Microsoft Internet Information Server or Microsoft Peer Web Services, any web browser can display the HTML output.

You can write ASP scripts using VBScript or JavaScript. Because these scripts interact with the ATK ActiveX server, they provide true object-oriented access to the data by means of the ATK object model. (In contrast, the ObjectStore Active Toolkit OLE DB provider uses a standard relational interface, rather than an object-oriented interface.) Thus, you can create and interact with ActiveX objects that implement behavior customized for your application.

Software requirements

To complete this tutorial, you need these software resources:

Resource	***Where to Find It***
Database	**\odi\ATK6.0\Examples\demodbs\carsdemo.db**
Visual Basic projects	**\odi\ATK6.0\Examples\Tutorial2**
Active Server Pages	Microsoft Internet Information Server 3.0 or later, or Microsoft Peer Web Services 3.0 or later

These sample applications were developed with Microsoft Visual InterDev 1.0. Although these sample applications were developed and tested with Microsoft Visual InterDev 1.0, you do not need it to run, build, or modify the source code.

In this chapter

In this chapter, you create ASP applications that use the ATK ActiveX server to retrieve tabular information from an ObjectStore database. This chapter contains these exercises:

Exercise	*Description*
Build an Application with ATK and ASP on page 23	Query an ObjectStore database, build a table, and display the table in HTML.
Use Inspector Instance Formats on page 31	Displays the same data view in HTML using different instance formats.
Display Multimedia Object Managers on page 36	Displays ObjectStore multimedia Object Managers using the ATK ActiveX server.
Display Tables of Tables on page 41	Display the data at many levels of abstraction, and allow the user to navigate to additional data.

Build an Application with ATK and ASP

Overview

This application uses ATK and ASP to query an ObjectStore database to retrieve ActiveX objects, build a table for display, and publish the output in HTML for display in any web browser.

Process

To build an application with ATK and ASP, follow these steps:

1 Create a data view to use in the application.

2 Customize the instance format.

3 Create the Active Server Page.

4 Create and access ATK ActiveX objects.

5 Test the application.

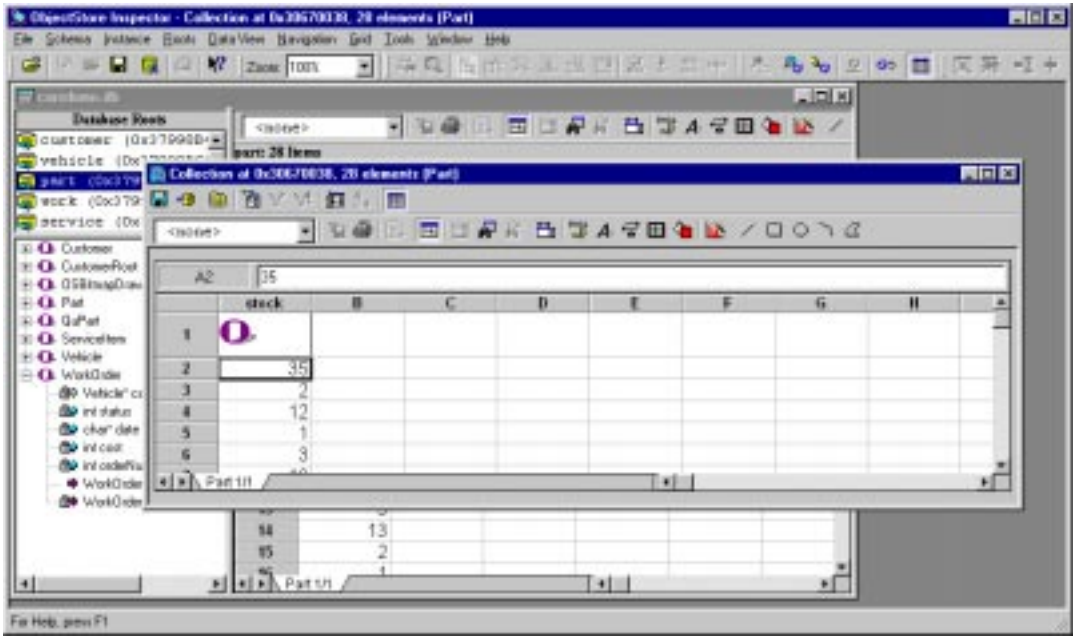
1 Create a data view to use in the application.

1 Start Inspector.

2 Open the sample database **\odi\ATK6.0\Examples\demodbs\carsdemo.db**.

3 Double-click on the **part** root name in the **Database Roots** pane of Inspector.

4 Select **Data View | Create** from the menu bar.



2 Customize the instance format.

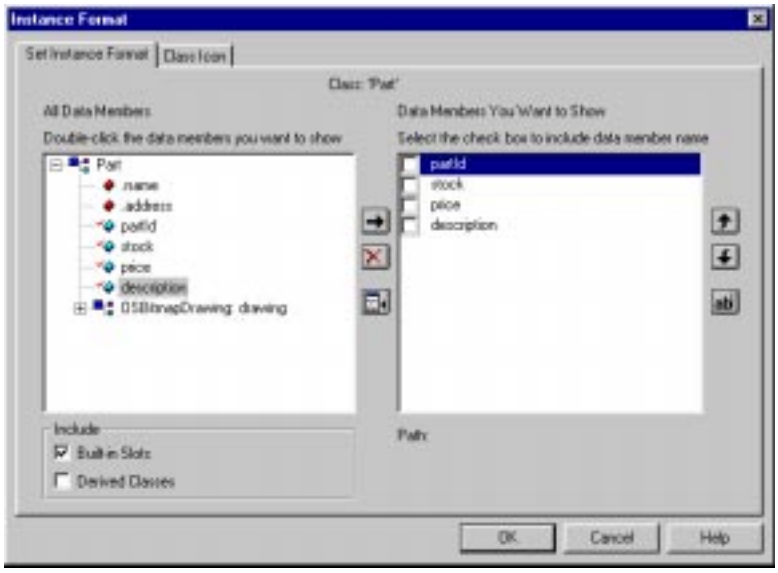
1 Right-click anywhere in the data view grid.

2 Select **Set Format of Class** from the shortcut menu.

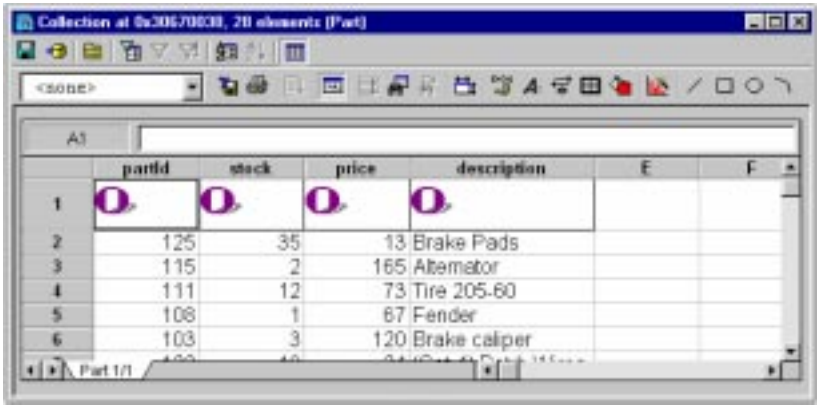
The Instance Format dialog box appears.

3 Modify the existing instance format so that it includes all data members in the **Part** class — **partId**, **price**, **stock**, and **description**.

Double-click on each data member to add it to the Data Members You Want to Show list box.



The data view is refreshed based on these changes to the instance format.



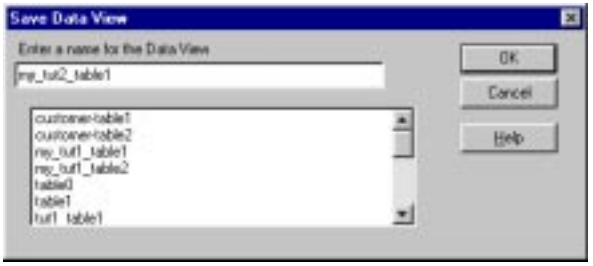
4 Select **Grid | Template | Save As** to save these changes as a grid template you can use again.

Name the grid template **my_tutorial2_1**.

Tip: In Inspector, instance format information is saved in *grid templates*. Instance format information is just one aspect of grid templates — grid templates include other information (fonts, formulas, and annotation, for example) that is not used by ATK.

5 Select **File | Save All**.

6 Name the new data view **my_tut2_table1**.



The metaknowledge for the **cardsdemo.db** database now contains the definition of the new table.

3 Create the Active Server Page.

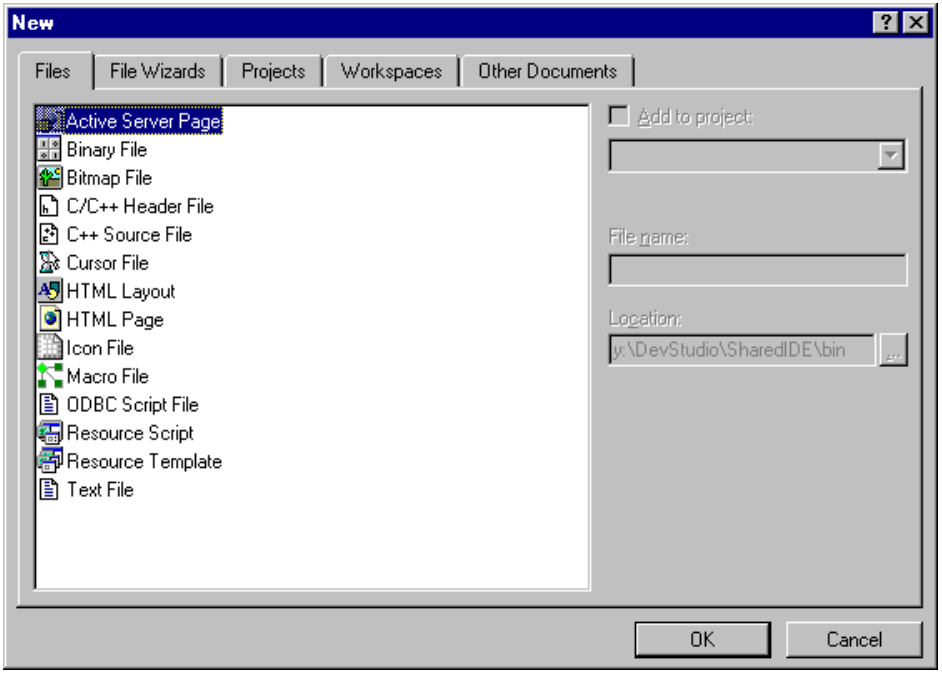
You can create an Active Server Page using Visual InterDev or a simple text editor. Procedures for using Visual InterDev are shown here.

Using Visual InterDev

To create an Active Server Page using Visual InterDev:

1 Start InterDev and select **File | New**.

2 On the **Files** sheet, select **Active Server Page** and click **OK**.



A new window appears, containing the Active Server Page.

```
<%@ LANGUAGE="VBSCRIPT" %>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual InterDev 1.0">
<META HTTP-EQUIV="Content-Type" content="text/html; charset=iso-8859-1">
<TITLE>Document Title</TITLE>
</HEAD>
<BODY>

<!-- Insert HTML here -->

</BODY>
</HTML>
```

Tip: If you are using a text editor to create an Active Server Page, create a file with an **.asp** extension that contains the text and format shown in the preceding window.

4 Create and access ATK ActiveX objects.

You can write VBScript or JavaScript code to create and access ATK ActiveX objects.

In VBScript

This VBScript code is very similar to the Visual Basic example in Chapter 1, A Visual Basic Application with ATK ActiveX Server:

```
<%
   On Error Resume Next
'Create the ATKKernel object
   Set theKernel=Server.CreateObject("ATKKernel.Document")
'Try opening the database
   Set theDatabase=theKernel.OpenDatabase("c:\odi\ATK6.0\Examples\demodbs\carsdemo.db")
   If Err.Number = 0 Then 'If no errors occurred...
'Open "my_tut2_table1"
   Set theDataView=theDatabase.GetDataView("my_tut2_table1")
   If Err.Number = 0 Then 'If no errors occurred...
      Set theTable= theDataView.GetATKTable 'Get the ATKTable object...
      WriteTable(theTable)              '...and print it
   Else  'Error condition
      Response.Write("Data view not found: " & Err.Description & "<BR>")
   End If
   Else   'Error condition
      Response.Write("Error opening database: " & Err.Description & "<BR>")
End If
%>
```

The **WriteTable** procedure can display any **ATKTable** object in HTML:

```
<SCRIPT LANGUAGE="VBScript" RUNAT=Server>
Sub WriteTable(theTable)
'Initialize the HTML table
   Response.Write("<TABLE BORDER=1>" & CHR(13) & CHR(10))
'Fill the column headings
   For Each colHeading in theTable.GetHeaders
      Response.Write("<TH><B><PRE>" & colHeading & "</PRE></B></TH>")
   Next
'Scan all the objects in the table
   while not theTable.IsEOT
'Get the tabular representation of each object
   For Each aRow in theTable.GetTabularRepresentation
'Each row in the representation is a row in the HTML table
      Response.Write("<TR>")
'Scan all the cells in the row
   For Each aCell in aRow
      Response.Write("<TD>" & aCell & "</TD>")
      Next
      Response.Write("</TR>" & CHR(13) & CHR(10))
```

```
    Next
'Jump to the next object in the table
    theTable.MoveNext
    wend
    Response.Write("</TABLE>" & CHR(13) & CHR(10))
End Sub
</SCRIPT>
```

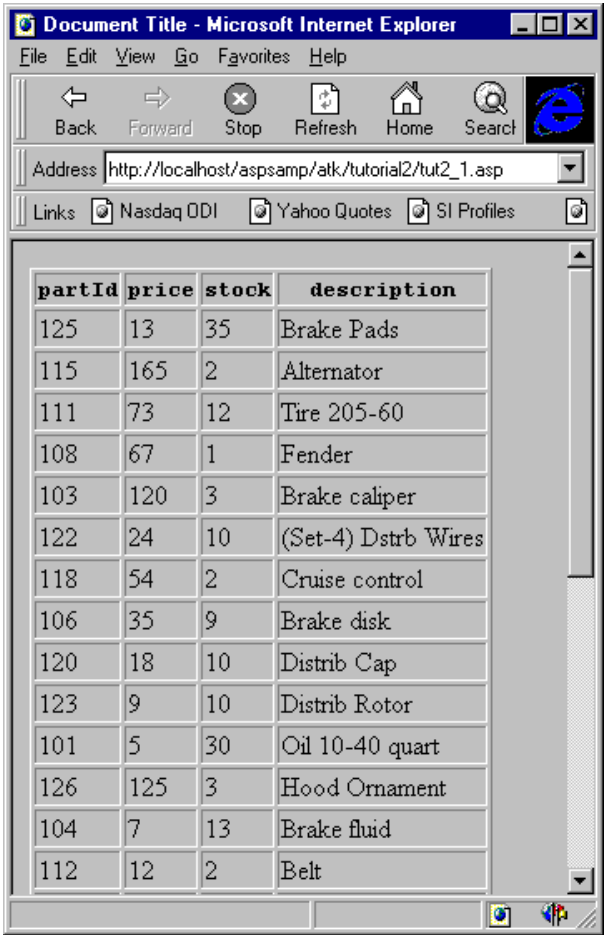
In JavaScript

You can also access an ATK ActiveX object through ASP using JavaScript. Refer to the **odi\ATK6.0\Examples\Tutorial3\tut2_2.asp** for an example of JavaScript code using the same data view.

5 Test the application.

1 Save the generated Active Server Page in the **INetPub\ASPSamp\ATK\Tutorial2** directory as **tut2_1.asp.**

2 In a web browser, open the URL **http://localhost/aspsamp/atk/tutorial2/tut2_1.asp**. Your browser window should display information similar to the following:



partId	price	stock	description
125	13	35	Brake Pads
115	165	2	Alternator
111	73	12	Tire 205-60
108	67	1	Fender
103	120	3	Brake caliper
122	24	10	(Set-4) Dstrb Wires
118	54	2	Cruise control
106	35	9	Brake disk
120	18	10	Distrib Cap
123	9	10	Distrib Rotor
101	5	30	Oil 10-40 quart
126	125	3	Hood Ornament
104	7	13	Brake fluid
112	12	2	Belt

Note that the four data members you defined in the instance format are included in this display.

Use Inspector Instance Formats

Overview

Instead of defining new data views every time you want to display different data, you can use different instance formats for the same data view. Instance formats describe the columns of data to display, the column order, and the column heading text.

Tip: Remember that instance formats are saved in grid templates in Inspector.

Process

To display data using a new instance format, follow these steps:

1 Define a new instance format.

2 Save the modified metaknowledge.

3 Modify the ASP code to refresh the metaknowledge.

4 Check the modified ASP code.

5 Change the ASP code to select the instance format.

1 Define a new instance format.

Using Inspector, define a new instance format for the class **Part**, and save the instance format in a collection grid.

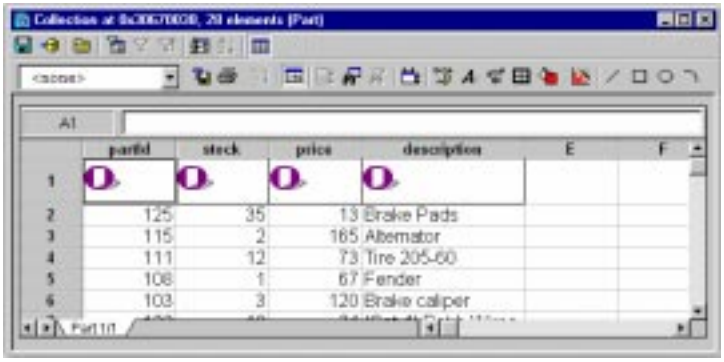
1 Start Inspector.

2 Open the **\odi\ATK6.0\demodbs\carsdemo.db** database.

3 Select **Data View | Open**.

4 Open the data view you saved in the previous exercise, **my_tut2_table1**.

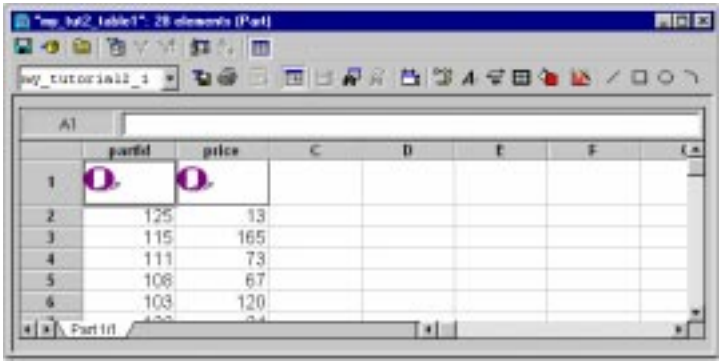
The Data View window displays the name of the grid template currently applied to the data view. In this case, it is **my_tutorial2_1**, which you saved in the previous example.



5 Right-click anywhere in the data view and select **Set Format of Grid Template** from the shortcut menu.

6 Remove the **stock** and **description** data members from the instance format.

The data view is refreshed based on your changes.



7 Select **Grid | Template | Save As** and save the instance format in the grid template **my_tutorial_2_2.**

2 Save the modified metaknowledge.

Select **File | Save All** to save the database. The metaknowledge for the database now associates the data view **my_tut2_table1** with the **my_tutorial2_2** grid template by default, which includes the new instance format.

However, if you reopen **tut2_1.asp**, it will behave as if the new instance format was never created or applied (returning the table with four columns). This is because ATK caches the metaknowledge of each database. In this case, ATK caches the metaknowledge of **\odi\ATK6.0\Examples\demodbs\carsdemo.db**. You need to modify the ASP code to refresh the metaknowledge.

3 Modify the ASP code to refresh the metaknowledge.

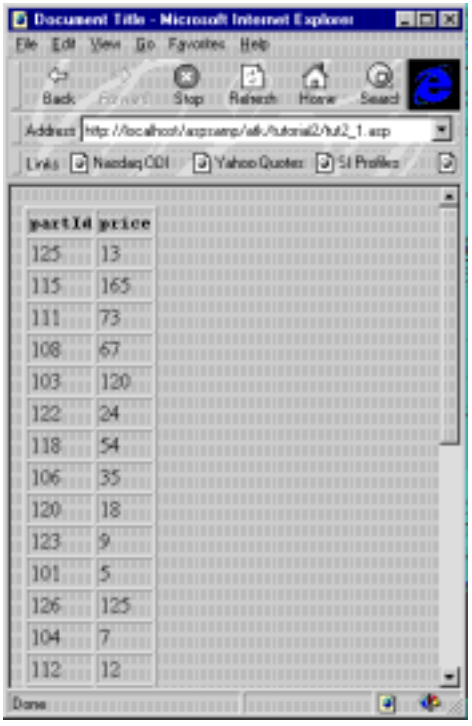
Modify **tut2_1.asp** so it forces ATK to reload the metaknowledge for all of the databases it has opened. In the first lines of the ASP code, call ATKKernel::ReloadMetaKnowledge.

```
<%
   On Error Resume Next
'Create the ATKKernel object
   Set theKernel=Server.CreateObject("ATKKernel.Document")
'Force ATKKernel to reload its metaknowledge cache
   theKernel.ReloadMetaknowledge
'Try opening the database
   Set theDatabase=theKernel.OpenDatabase("c:\odi\ATK6.0\Examples\demodbs\carsdemo.db")
...
%>
```

4 Check the modified ASP code.

In a web browser, open the URL **http://localhost/aspsamp/atk/tutorial2/tut2_1.asp** to display the result, which uses the instance format saved in the **my_tutorial2_2**

grid template. Your browser window should display information similar to the following:



Note that the display now includes only two data members: **partId** and **price**.

5 Change the ASP code to select the instance format.

You can modify the application to choose the instance format from among those you have saved with any grid template in the database. **GetATKTable** accepts an optional argument in which you can specify the name of the instance format you want to retrieve. Refer to ATKDataView::GetATKTable in the *ObjectStore Active Toolkit Reference* for details.

1 In **tut2_1.asp**, modify the **GetATKTable** call using the format **<ClassName>::<InstanceFormatName>**:

```
Set theTable = theDataView.GetATKTable("Part::my_tutorial2_1")
WriteTable(theTable)
```

This call forces ATK to format the selected data view using the **Part::my_tutorial2_1** instance format saved with the **my_**

tutorial2_1 grid format. (You can specify any valid **Part** instance format that you have saved with an Inspector grid template.)

2 Remove the **ATKKernel::ReloadMetaknowledge** call. Because you have not modified the database metaknowledge, refreshing it is unnecessary.

3 Reload the web browser page. The table now displays four columns, including the **stock** and **description** data members.

Display Multimedia Object Managers

Overview

Although you can access multimedia data through the ObjectStore Active Toolkit OLE DB provider, you must do so through a standard OLE DB or ADO (Active Data Objects) interface. However, an ASP script that queries the ATK ActiveX server can directly access the multimedia Object Manager instances, retrieve the data, and display it in a web browser.

Process

To display multimedia object managers, follow these steps:

1 Determine how to access the image Object Manager instances in the database.

2 Write the ASP code to display multimedia Object Managers.

1 Determine how to access the image Object Manager instances in the database.

The way in which you access image Object Manager instances depends on whether or not they are connected to a root.

- If the images are contained in a collection that is attached to a root, you can access the images through the ATK ActiveX server by specifying the root name.
- If the images are not connected to a root, you must define a data view to access them.

Use Inspector

You can use Inspector to browse multimedia Object Manager instances to determine whether or not they are attached to a root.

1 Start Inspector.

2 Open the **\odi\ATK6.0\Examples\demodbs\extrademo.db** database.

The **IMAGES** root contains an ObjectStore collection of **osmmVirageImage** instances.

3 Double-click on the **IMAGES** root to populate the instance pane.

4 Double-click on an element in the collection to see the stored image.

Because the images are contained in a collection that is attached to a root, there is no need to define a data view to access them. You can access the images through the ATK ActiveX server by specifying the root name.

2 Write the ASP code to display multimedia Object Managers.

The structure of an HTML page requires two Active Server Pages:

- The first page opens the database, retrieves the root, and builds a table containing IMG SRC tags that point to the second page.
- The second page retrieves the reference to the image and dumps the actual data in the image to the HTML page.

Using Microsoft Visual InterDev or any text editor:

1 Create the first Active Server Page with this code:

```
<%
   On Error Resume Next
   Set theKernel=Server.CreateObject("ATKKernel.Document")
'Open the extrademo.db database
   Set theDatabase=theKernel.OpenDatabase("c:\odi\ATK6.0\Examples\demodbs\extrademo.db")
   If Err.Number = 0 Then 'If no errors occurred...
'Retrieve the "exhibit_root" root
   Set theRoot=theDatabase.GetRoot("IMAGES")
'Get the ATKReference of the object associated with the root
   Set rootValue=theRoot.GetATKReference
'Get the ATKReferences representing the collection associated with the root
   Set rootValues=rootValue.GetCollectionItems
'Initialize the HTML table
      Response.Write("<TABLE BORDER=1>")
      Response.Write("<TH><B>Images</B></TH>")
'Loop on all the Object Managers in the collection
   For Each anOM in rootValues
      Response.Write("<TR><TD>")
'Build an URL pointing to tut2_om2.asp, containing the code that dumps an OM
      Response.Write("<IMG SRC=""tut2_om2.asp?dumpedReference=" & _
         Server.URLEncode(anOM.GetReference) & """>")
      Response.Write("</TD></TR>")
   Next
      Response.Write("</TABLE>" & CHR(13) & CHR(10))
   Else
      Response.Write("Error opening database: " & Err.Description & "<BR>")
   End If
%>
```

2 Save this page in the **INetPub\ASPSamp\ATK\Tutorial2** directory as **tut2_om1.asp**.

3 Create the second Active Server Page to retrieve the image. If you are using Visual InterDev, remove the default InterDev code. The output of this page must contain the type and the actual data from the multimedia Object Manager.

This is the code for the second page:

```
<%@ LANGUAGE="VBSCRIPT" %>
<%
   On Error Resume Next
   Set theKernel=Server.CreateObject("ATKKernel.Document")
'Resolve the reference passed in the "dumpedReference" HTML variable
   Set theObject=theKernel.ResolveReference( _
      Request.QueryString("dumpedReference"))
   If theObject.IsObjectManager Then  'If the object is an Object Manager...
'Get the ATKObjectManager object
   Set theOM=theObject.GetATKObjectManager
'Store its mime type in the Response.ContentType field
      Response.ContentType=theOM.GetOMMimeType
'Write the OM bits
      Response.BinaryWrite(theOM.GetDataArray)
   End If
%>
```

4 Save the Active Server Page in the **INetPub\ASPSamp\ATK\Tutorial2** directory as **tut2_om2.asp**.

5 In a web browser, open the URL **http://localhost/aspsamp/atk/tutorial2/tut2_om1.asp**. The information should be similar to that shown here:

Display Tables of Tables

Overview

Typically, when you extract HTML tables from a database, you want to display the data at many levels of abstraction, and allow the user to navigate to additional data. The application you create in this section demonstrates how you can use ATK ActiveX server to create a table that displays all work orders in a database and, upon the user's request, detail all the service items for a particular work order.

Process

To display tables of tables, follow these steps:

1 Check the data view in Inspector.

2 Check the data view format.

3 Build the ASP code to display a table of tables.

4 Test the application.

1 Check the data view in Inspector.

This example reuses a prototype of the **my_tut1_table2** data view you defined in Chapter 1, A Visual Basic Application with ATK ActiveX Server.

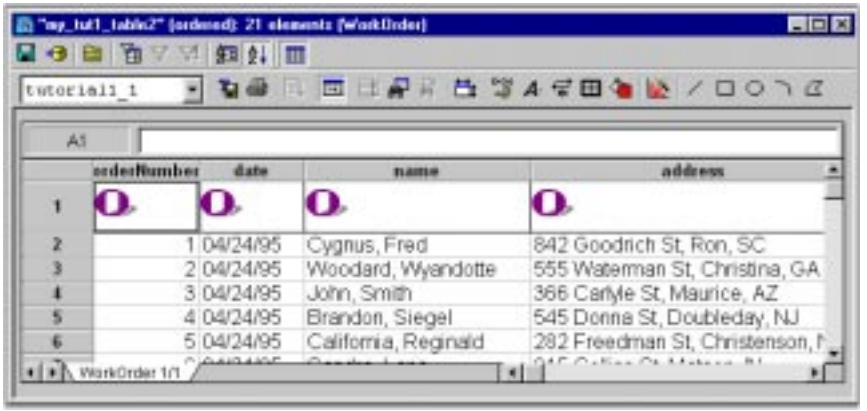
2 Check the data view format.

1 Start Inspector.

2 Open the **\odi\ATK6.0\Examples\demodbs\carsdemo.db** database.

3 Select **Data View | Open**.

4 Select the **my_tut1_table2** data view.



Use this data view to build the main table for display in HTML. You want to design the application so a user accessing this information in a web browser can click on the **orderNumber** column to get the details of the relation, called **items**, which links the **WorkOrder** class to the **ServiceItem** class.

5 Click on the **service** root in Inspector to preview the instance format ATK uses to display **ServiceItem** instances.

3 Build the ASP code to display a table of tables.

In the example Display Multimedia Object Managers on page 36, the Active Server Page simply displays a collection of images in a web browser. However, this example contains additional functionality that allows the user to click on an item in the first column of the table and navigate that item's relationship to more detailed data.

1 Add a link to the first column so the user can navigate. In **tut2_1.asp**, change the **WriteTable** procedure so it contains this code:

```
For Each aRow in theTable.GetTabularRepresentation
    Response.Write("<TR>")
    isFirstColumn=true 'used to modify the first column
    For Each aCell in aRow
        If isFirstColumn Then  'is this the first column?
'if it is the first column, navigate the "items" relation
    Set theRelatedItems=theTable.GetObject.GetSlotValue("items")
'then use the returned ATKReference to build the URL of the link;
'we send the dumped reference to the tut2_nav2 page
    Response.Write("<TD><A HREF=""tut2_nav2.asp?dumpedReference=" _
        & Server.URLEncode(theRelatedItems.GetReference) & """>" _
        & aCell & "</A></TD>")
    Else  'this isn't the first column
    Response.Write("<TD>" & aCell & "</TD>")
    End If
    isFirstColumn=false  'no more the first column
        Next
        Response.Write("</TR>" & CHR(13) & CHR(10))
    Next
```

2 Save this page in the **INetPub\ASPSamp\ATK\Tutorial2** directory as **tut2_nav1.asp**.

3 The second Active Server Page must decode the **dumpedReference** value and build the **ATKTable** object.

Use this code for the second page:

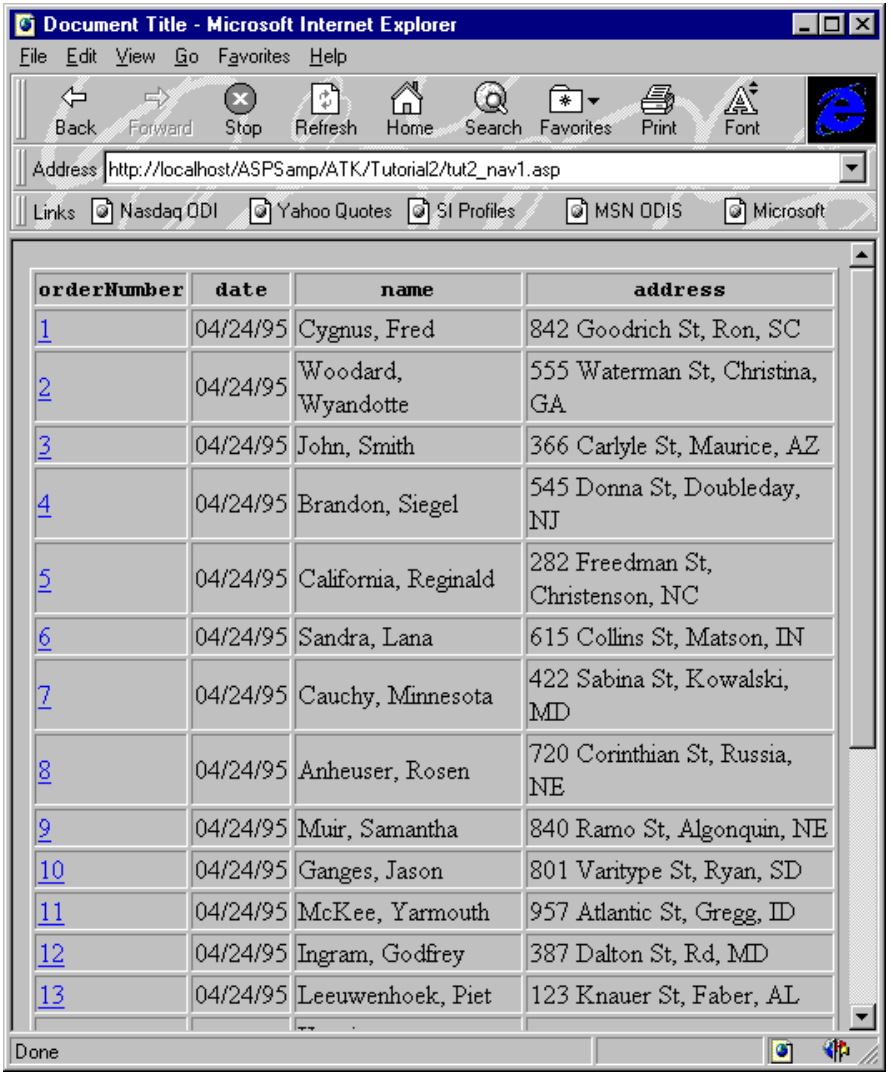
```
<%
    On Error Resume Next
    Set theKernel=Server.CreateObject("ATKKernel.Document")
'Resolve the reference passed in the "dumpedReference" HTML variable
    Set theObject=theKernel.ResolveReference( _
        Request.QueryString("dumpedReference"))
'If the object is a collection
    If theObject.IsCollection Then
'write the table gotten through an ATKReferences object
        WriteTable(theObject.GetCollectionItems.GetATKTable("ServiceItem"))
    End If
%>
```

The **WriteTable** procedure is the same one used in **tut2_1.asp**. Note that the **GetATKTable** call specifies the name of the class whose instances are contained in the ObjectStore collection that is being displayed. Thus, ATK formats the **ATKTable** object using the default **ServiceItem** instance format.

4 Save the second Active Server Page in the **INetPub\ASPSamp\ATK\Tutorial2** directory as **tut2_nav2.asp**.

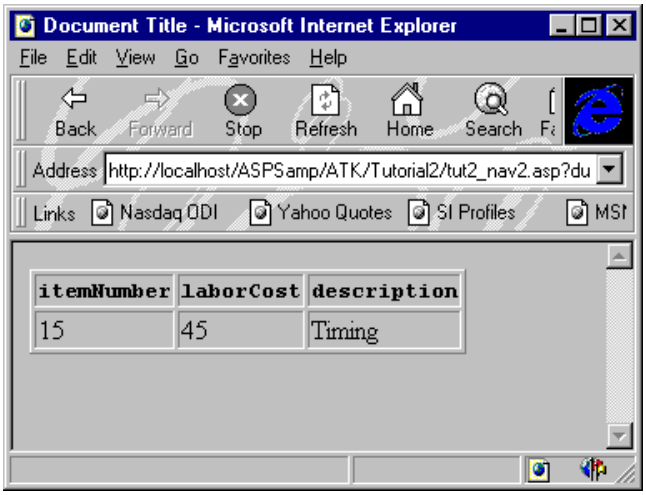
4 Test the application.

1 In a web browser, open the URL **http://localhost/aspsamp/atk/tutorial2/tut2_nav1.asp**. The information displayed should be similar to that shown here:

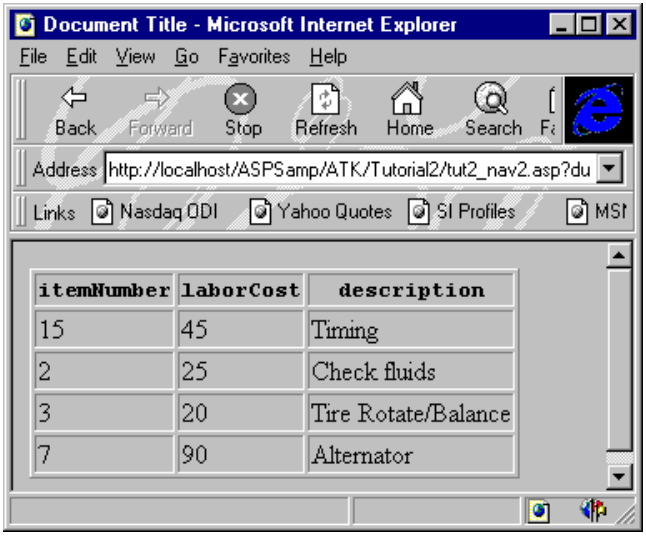


orderNumber	date	name	address
1	04/24/95	Cygnus, Fred	842 Goodrich St, Ron, SC
2	04/24/95	Woodard, Wyandotte	555 Waterman St, Christina, GA
3	04/24/95	John, Smith	366 Carlyle St, Maurice, AZ
4	04/24/95	Brandon, Siegel	545 Donna St, Doubleday, NJ
5	04/24/95	California, Reginald	282 Freedman St, Christenson, NC
6	04/24/95	Sandra, Lana	615 Collins St, Matson, IN
7	04/24/95	Cauchy, Minnesota	422 Sabina St, Kowalski, MD
8	04/24/95	Anheuser, Rosen	720 Corinthian St, Russia, NE
9	04/24/95	Muir, Samantha	840 Ramo St, Algonquin, NE
10	04/24/95	Ganges, Jason	801 Varitype St, Ryan, SD
11	04/24/95	McKee, Yarmouth	957 Atlantic St, Gregg, ID
12	04/24/95	Ingram, Godfrey	387 Dalton St, Rd, MD
13	04/24/95	Leeuwenhoek, Piet	123 Knauer St, Faber, AL

2 Click **orderNumber 1** to display its related **item**:



3 Click **orderNumber 13** to display its details:



Summary

In this chapter, you built simple Active Server Page applications using data retrieved by the ATK ActiveX server. By using more of the classes and methods in the ATK object model, you can build much more sophisticated applications.

You can also implement these examples in a context other than the web, such as Visual Basic.

Chapter 3
Accessing ATK OLE DB Provider from Active Server Page Applications

Introduction

The OLE DB provider uses a standard interface between an application and a data source that allows you to specify a query using SQL syntax. The queries return tables of data in rows and columns.

With Active Server Pages, you can write *server side* scripts in VBScript or JavaScript that create and manipulate ActiveX objects, which implement the application's customized behavior. These scripts instantiate COM objects and access the ObjectStore Active Toolkit OLE DB provider through ADO (ActiveX Data Objects), which is a simple, standard object model. The scripts then return data from the ObjectStore Active Toolkit OLE DB provider in standard HTML. Any web browser can process the data from Active Server Pages that use ADOs.

You might prefer to use the ObjectStore Active Toolkit OLE DB provider instead of ATK ActiveX server — because it provides a standard interface, there is no need to become familiar with another object model. The Active Toolkit OLE DB data source is accessible through the standard ADO interface and provides essentially the same functionality as the ATK ActiveX server (that is, it generates a table of data from an ObjectStore database). The ObjectStore Active Toolkit OLE DB source's ability to provide nested row sets from ADO 1.5 gives you flexibility in accessing the underlying ObjectStore database.

By integrating ASP with the ObjectStore Active Toolkit OLE DB provider, you can leverage the strengths of both technologies. ASP lets you access an object-oriented database through a standard interface.

Software requirements

To complete the exercises in this chapter, you need these software resources:

Resource	*Where to Find It*
Database	**\odi\ATK6.0\Examples\demodbs\carsdemo.db**
Active Server Page source code	**\odi\ATK6.0\Examples\Tutorial3**
Active Server Pages	Microsoft Internet Information Server 3.0 or later, or Microsoft Peer Web Services 3.0 or later

Although these sample applications were developed with Microsoft Visual InterDev 1.0, you do not need it to run, build, or modify the source code.

In this chapter

In this chapter, you create Active Server Page applications that use the ATK OLE DB data source to retrieve tabular information from an ObjectStore database. This chapter contains the following exercises:

Exercise	*Description*
Build an Application with ATK, ASP, and ADO on page 50	Query an ObjectStore database, build a table of data, and display the table in HTML.
Customize a Data View in ADO on page 55	Display specific fields from a data view, and let the user navigate among data items in a web browser.
Implement Explicit Navigation in ADO on page 57	Query the database and build a table of data that specifically names the classes among which a user can navigate.
Display Multimedia Object Managers Using ADO on page 62	Retrieve multimedia objects from an ObjectStore database for display in a web browser.

Write the ASP-ADO Code to Show Multimedia Object Managers on page 64	Scan the database for an image and store it. Use the application to retrieve and display the image.

Build an Application with ATK, ASP, and ADO

Overview

This application uses ATK, ASP, and ADO to query an ObjectStore database, build a table, and display the table in HTML. ATK contains the ObjectStore Active Toolkit OLE DB provider, which ADO or any other OLE DB consumer can access. Even if VBScript and JavaScript cannot directly access the OLE DB interface, they can still access OLE DB sources through ADO.

Process

To build an application with ATK, ASP, and ADO, follow these steps:

1 Create a new data view to use in the application.

2 Create the Active Server Page.

3 Access the data source through ADO.

4 Test the application.

1 Create a new data view to use in the application.

1 Start Inspector.

2 Open the sample database **\odi\ATK6.0\Examples\demodbs\carsdemo.db**.

3 Click on the **service** root name in the **Database Roots** pane of Inspector.

4 Select **Data View | Create**.

An untitled Data View window appears.

5 Select **File | Save All**.

The Save Data View dialog box appears.

6 Name the new data view **my_tut3_table1** and click **OK**.

The metaknowledge for the **cardsdemo.db** database now contains the definition of the new table.

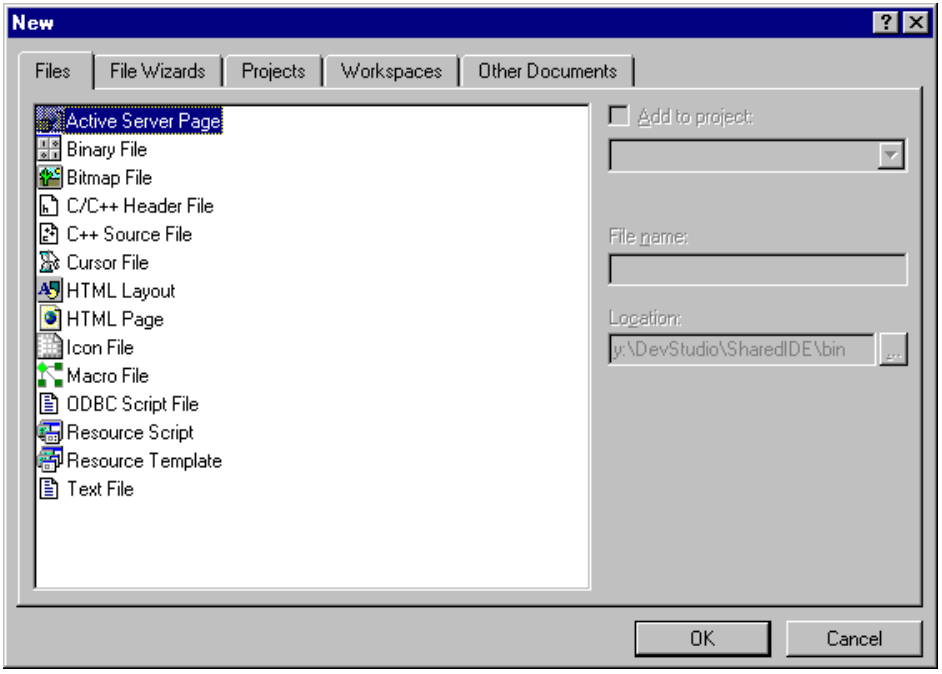
2 Create the Active Server Page.

Using Visual InterDev

You can create an Active Server Page using Visual InterDev or a simple text editor. Procedures for using Visual InterDev are shown here.

1 If you are using Visual InterDev, start that application and select **File | New**.

2 On the **Files** sheet of the New dialog box, select **Active Server Page** and click **OK**.



A new window appears; it displays the Active Server Page.

```
<%@ LANGUAGE="VBSCRIPT" %>

<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual InterDev 1.0">
<META HTTP-EQUIV="Content-Type" content="text/html; charset=iso-8859-1">
<TITLE>Document Title</TITLE>
</HEAD>
<BODY>

<!-- Insert HTML here -->

</BODY>
</HTML>
```

Tip: If you are using a text editor to create an Active Server Page, create a file with an **.asp** extension that contains the text and format shown in the preceding window.

3 Access the data source through ADO.

You can write VBScript or JavaScript code to create and access ATK ActiveX objects.

In VBScript

This VBScript example creates an ADO **RecordSet** object based on the **my_tut3_table1** data view and displays it in HTML.

```
<%
   On Error Resume Next
'Create the ADO connection
   Set adoConnection = Server.CreateObject("ADODB.Connection")
'Open the ADO - OLE DB connection; here you must specify the name
'of the ATK OLE DB Provider and the name of the database
   Call adoConnection.Open( _
      "provider=ObjectStore Active Toolkit OLE DB Provider;" & _
      "data source=c:\odi\ATK6.0\Examples\demodbs\carsdemo.db","","")
   If Err.Number<>0 Then  'Something went wrong
      Response.Write("Error: " & Err.Description)
   Else  'Connection has been correctly opened
'Create an ADO RecordSet
   Set adoRS = Server.CreateObject("ADODB.RecordSet")
'And open data view "my_tut3_table1" using the previously opened ADO connection
   Call adoRS.Open("my_tut3_table1", adoConnection)
   If Err.Number<>0 Then  'Something went wrong
      Response.Write("Error: " & Err.Description)
   Else 'RecordSet has been correctly opened
'Generate the HTML output for the opened RecordSet
      WriteRecordSetTable(adoRS)
      End If
   End If
%>
```

The **WriteRecordSetTable** procedure displays any ADO **RecordSet** object in HTML:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub WriteRecordSetTable(anADORS)
   On Error Resume Next
'Initialize the HTML table
   Response.Write("<TABLE BORDER=1>")
'Generate the column headings
   For Each colHeading in anADORS.fields
      Response.Write("<TH><B><PRE>" & colHeading.Name & "</PRE></B></TH>")
   Next
'Scan all the records in the ADO RecordSet
   Do While Not anADORS.EOF
      Response.Write("<TR>")
'Retrieve all the fields in the current row
   For Each aField in anADORS.fields
```

```
'Output the cell content
        Response.Write("<TD>" & aField.Value & "</TD>")
    Next
        Response.Write("</TR>")
'Move to the next row in the ADO RecordSet
        anADORS.MoveNext
    Loop
    Response.Write("</TABLE>")
End Sub
</SCRIPT>
```

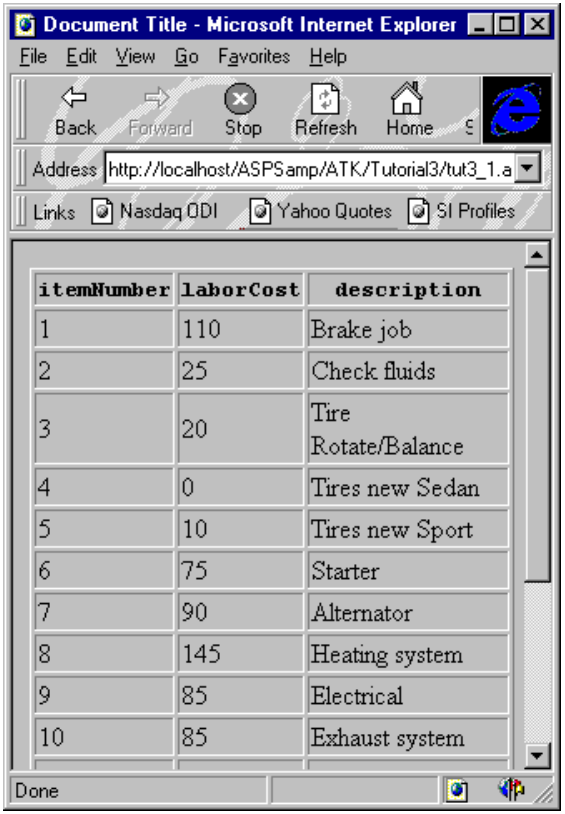
In JavaScript

You can also access an Active Toolkit OLE DB data source through ADO using JavaScript. Refer to **odi\Atk\Examples\Tutorial3\tut3_2.asp** for an example that uses the same data view.

4 Test the application.

1 Save the generated Active Server Page in the **INetPub\ASPSamp\ATK\Tutorial3** directory as **tut3_1.asp**.

2 In a web browser, open the URL **http://localhost/aspsamp/atk/tutorial3/tut3_1.asp**.



itemNumber	laborCost	description
1	110	Brake job
2	25	Check fluids
3	20	Tire Rotate/Balance
4	0	Tires new Sedan
5	10	Tires new Sport
6	75	Starter
7	90	Alternator
8	145	Heating system
9	85	Electrical
10	85	Exhaust system

Optimization

You can optimize this application by using ASP sessions to cache the ADO connection and **RecordSet** objects. Refer to **odi\ATK6.0\Examples\WebADODemo** for an example.

Customize a Data View in ADO

Overview

You can customize a data view in ADO using the SQL syntax accepted by the ObjectStore Active Toolkit OLE DB provider. For example, instead of displaying the fields contained in the data view **my_tut3_table1**, you can display the **itemNumber** and **description** data members of the **ServiceItem** class, and navigate the **orders** relation and display the **orderNumber** of the related **WorkOrder** instance.

For more information

Refer to Chapter 3, Active Toolkit OLE DB Provider, in the *ObjectStore Active Toolkit Reference*, for more information about how ATK works with SQL code.

Process

To customize a data view in ADO, follow these steps:

1 Select the data members to display.

2 Specify the column headings and the order of the data.

1 Select the data members to display.

To select the data members to display, modify the VBScript line that opens the ADO **RecordSet** object in **tut3_1.asp** as shown here:

```
Call adoRS.Open( _
"SELECT itemNumber, description, orders#orderNumber FROM my_tut3_table1", _
adoConnection)
```

This SQL command instructs ATK to display three data members from the items contained in the **my_tut3_tab1** data view: **itemNumber**, **description**, and **orders#orderNumber**. The last item contains the navigation of the **orders** relation.

Because **orders** is a one-to-many relationship, this SQL command might generate multiple lines for every **ServiceItem** contained in the source collection. For example, the **ServiceItem** whose **itemNumber** is **7** has two associated **WorkOrder** instances, as shown above in the generated table.

2 Specify the column headings and the order of the data.

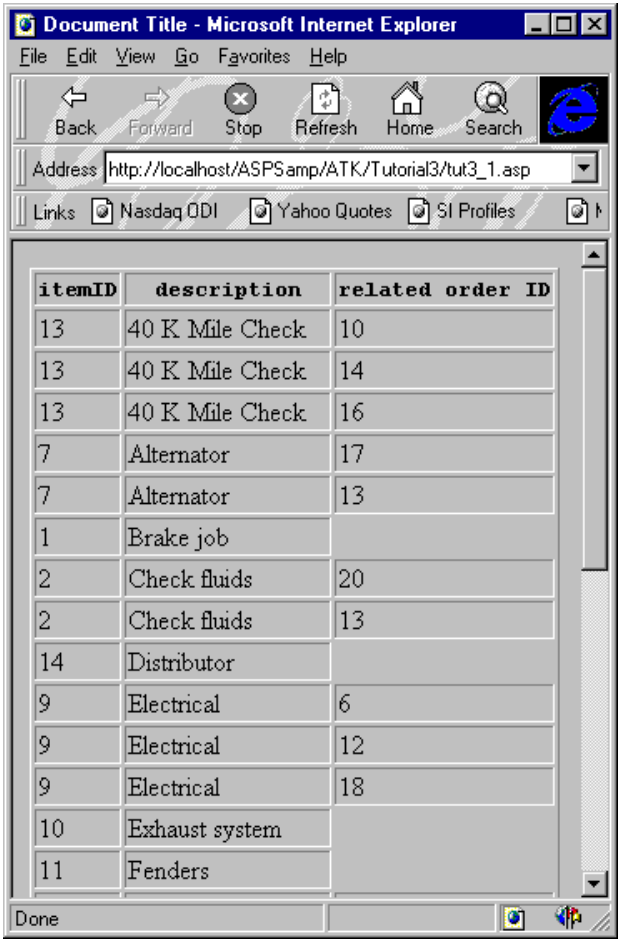
In this step, you specify a more complex SQL command to

- Label the column headings differently
- Order the collection according to the **description** data member of the **ServiceItem** class

You use the following SQL command:

SELECT itemNumber as itemID, description, orders#orderNumber AS 'related order ID' FROM my_tut3_table1 ORDER BY description

This is the result of the query:



itemID	description	related order ID
13	40 K Mile Check	10
13	40 K Mile Check	14
13	40 K Mile Check	16
7	Alternator	17
7	Alternator	13
1	Brake job	
2	Check fluids	20
2	Check fluids	13
14	Distributor	
9	Electrical	6
9	Electrical	12
9	Electrical	18
10	Exhaust system	
11	Fenders	

Implement Explicit Navigation in ADO

Overview

You can build a table that allows the user of an application to navigate from one item to another explicitly. That is, instead of using implicit navigation to access one class through another class, the application accesses each class directly by naming the class. When you navigate with ADO, do not use **Field.Value** as an integer value. Instead, use it to retrieve a **RecordSet** that represents the elements related to the object displayed in the current row.

When you extract HTML tables from a database, you usually want to display the data at many levels of abstraction, and let the user navigate to additional data.

The application you build in this section demonstrates how you can use ADO 1.5 and later and the ObjectStore Active Toolkit OLE DB provider to create a table that displays all the service items in a database. The user can click on a particular service item to display all of its work orders; this is *explicit navigation*. To navigate explicitly using ADO, you must use *nested* **RecordSet**s.

Using ADO 1.5

ADO 1.5 and later support *chaptered row sets*. In this case, the **Field.Value** objects from the columns of data that provide navigation are themselves **RecordSet** objects representing the elements related to the object displayed in the current row. When an ADO 1.5 or later consumer accesses the ObjectStore Active Toolkit OLE DB provider, the provider automatically implements explicit navigation through chaptered row sets.

Process

To implement explicit navigation, follow these steps:

1 Modify the SQL statement that opens the RecordSet.

2 Modify the procedure that displays the RecordSet.

3 Write a new page that displays the navigated RecordSets.

4 Test the application.

1 Modify the SQL statement that opens the RecordSet.

The **SELECT** commands you have used so far included only *implicit* navigation among classes. (That is, they handle data members of related classes as if they are data members of the displayed class.) ATK accepts additional SQL syntax, with which you can specify nested **SELECT** commands. Every nested SQL command is translated into nested **RecordSet** objects.

Modifying the SQL

Suppose you want to display the **ServiceItem** instances of the data view **my_tut3_table1** and provide a link for each row that describes the related **WorkOrder** instances. In **tut3_1.asp**, use this SQL command:

```
SELECT itemNumber, description, {SELECT orderNumber, car#model FROM orders} FROM tut3_
table1
```

The last column specified in the first **SELECT** command is another nested **SELECT** directive, in which you specify the data members of the **WorkOrder** class you want to display (**orderNumber** and the implicit navigation **car#model**) and the name of the **ServiceItem** data member that implements the relation with the **WorkOrder** class (**orders**).

2 Modify the procedure that displays the RecordSet.

Before running this SQL command, modify the **WriteRecordSetTable** procedure that you created previously.

1. Copy the **tut3_1.asp** file to **tut3_nav1.asp**.
2. To handle the links between the main table and the related subtables, use the **Session ASP** object.

   This is the modified and commented code for ADO 1.5 or later:

```
Sub WriteRecordSetTable(anADORS)
   On Error Resume Next
'Remove any previous navigation information
   Session("Navigations")=Empty
'Store the main RecordSet
   Set Session("MainRS")=anADORS
'Create an array that will store the navigated RecordSets
   Dim  navigatedRS()
   ReDim  navigatedRS(anADORS.RecordCount)
'Initialize the HTML table
      Response.Write("<TABLE BORDER=1>")
'Generate the column headings
   For Each colHeading in anADORS.fields
   Response.Write("<TH><B><PRE>" & colHeading.Name & "</PRE></B></TH>")
   Next
```

```
'Scan all the records in the ADO RecordSet and keep track of the row number
    rowNumber=0
    anADORS.MoveFirst
    Do While Not anADORS.EOF
        Response.Write("<TR>")
'Retrieve all the fields in the current row
    For Each aField in anADORS.fields
if aField.Type=136, this is a chaptered rowset
        If aField.Type=136 Then
'Store the navigated RecordSet
            Set navigatedRS(rowNumber)=aField.Value
'Retrieve the number of records in the navigated RecordSet
            recordCount=navigatedRS(rowNumber).RecordCount
            If recordCount>0 Then
'Create the link to the second page iff recordCount>0
            Response.Write("<TD><A HREF=""tut3_nav2.asp?navPosition=" _
                & rowNumber & """>" & recordCount & " related items</A>")
            Else
                Response.Write("<TD>No Related Items</TD>")
            End If
        Else  'The usual cell content
            Response.Write("<TD>" & aField.Value & "</TD>")
        End If
    Next
        Response.Write("</TR>")
'Move to the next row in the ADO RecordSet
    anADORS.MoveNext
    rowNumber=rowNumber+1
    Loop
        Response.Write("</TABLE>")
'Store the array of navigated RecordSets in the Session
    Session("Navigations")=navigatedRS
End Sub
```

3 Write a new page that displays the navigated RecordSets.

1 In a text editor, create a new file that contains this code:

```
<%
    On Error Resume Next
'Retrieve the position of the navigated RS in the array
    navPosition=Request.QueryString("navPosition")
'Retrieve the array of RecordSets
    navArray=Session("Navigations")
'Get the adoRS located at the navPosition index in the array
    Set adoRS=navArray(navPosition)
'Write the navigated adoRS
    WriteRecordSetTable(adoRS)
    Set adoRS=Nothing
```

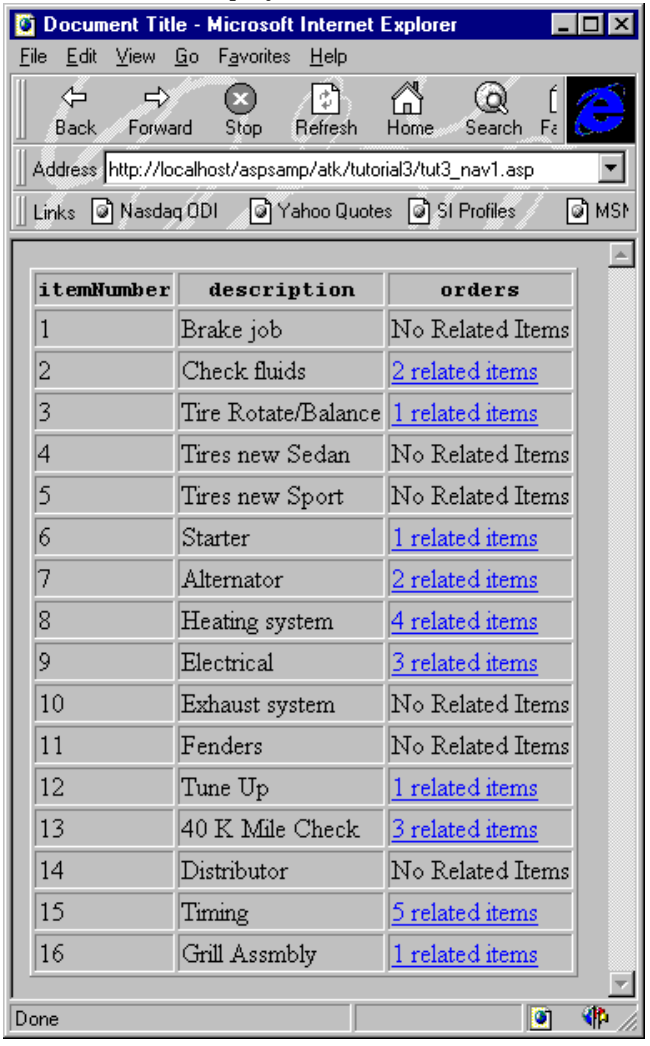
```
%>
```

WriteRecordSetTable is the simple procedure you already wrote in **tut3_1.asp**; you can copy it to **tut3_nav2.asp**.

2 Save the file in the **InetPub\ASPSamp\ATK\Tutorial3** directory as **tut3_nav2.asp**.

4 Test the application.

1 With a web browser, open the Active Server Page **http://localhost/ASPSamp/ATK/Tutorial3/tut3_nav1.asp**. Your browser should display information similar to that shown here:



itemNumber	description	orders
1	Brake job	No Related Items
2	Check fluids	2 related items
3	Tire Rotate/Balance	1 related items
4	Tires new Sedan	No Related Items
5	Tires new Sport	No Related Items
6	Starter	1 related items
7	Alternator	2 related items
8	Heating system	4 related items
9	Electrical	3 related items
10	Exhaust system	No Related Items
11	Fenders	No Related Items
12	Tune Up	1 related items
13	40 K Mile Check	3 related items
14	Distributor	No Related Items
15	Timing	5 related items
16	Grill Assmbly	1 related items

2 To navigate, click an **orders** link that displays one or more related items:

The browser content changes to reflect the order you clicked.

Display Multimedia Object Managers Using ADO

Overview

The ObjectStore Active Toolkit OLE DB provider fully supports multimedia Object Managers. From an Active Server Page, you can use ADO and ATK to access multimedia Object Managers stored in an ObjectStore database.

This sample application is an ASP page that displays a data view consisting of multimedia Object Managers that store images.

Process

To display multimedia Object Managers using ADO, check the data view that contains the multimedia images:

1 Start Inspector.

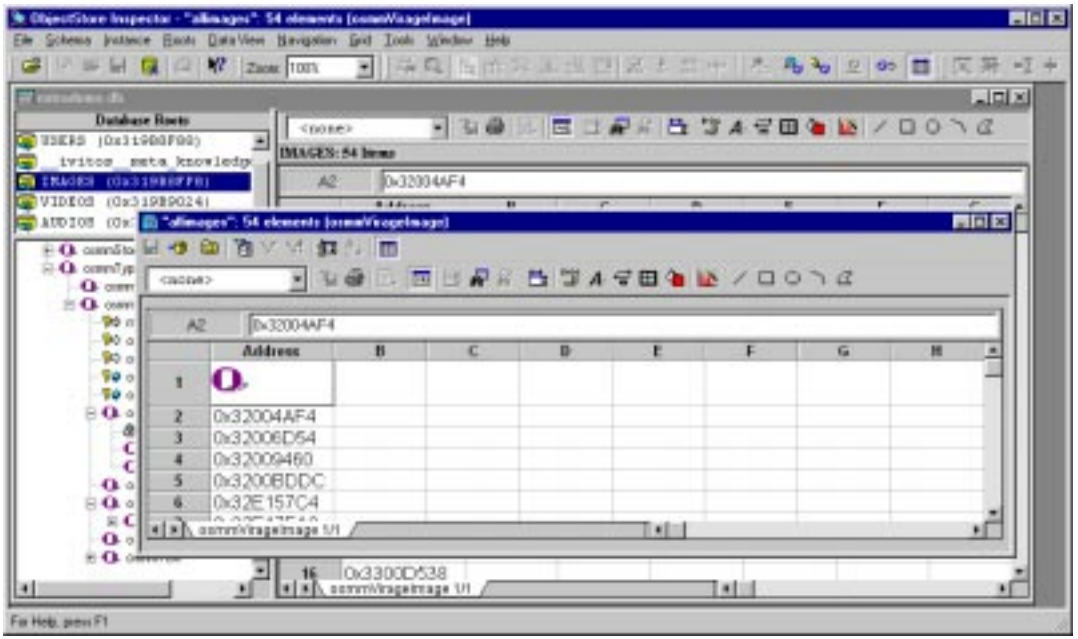
2 Open the sample database **c:\odi\ATK6.0\Examples\demodbs\extrademo.db**.

3 Select **Data View | Open**.

The Open Data View dialog box appears.

4 Select the **allimages** data view and click the **OK** button.

The Data View window appears with the **images** data view displayed.



The data view contains a collection of instances of the Image Object Manager, **osmmViragelmage**.

5 To check the content of the data view, double-click on an element to display it:

The **extrademo.db** database contains a data view called **allimages** that contains Object Managers. Use this data view to display images on an ASP page.

Write the ASP-ADO Code to Show Multimedia Object Managers

Overview

This example is similar to the explicit navigation example described in Implement Explicit Navigation in ADO on page 57. The code sample described in this section scans the **RecordSet** containing the images, stores the image data in an array referenced by a **Session ASP** object, and then uses this object in the referred page to draw the image.

Process

To show multimedia object managers, follow these steps:

1 Create a new ASP page to retrieve the data.

2 Create another ASP page to display the data.

3 Test the application.

1 Create a new ASP page to retrieve the data.

1 Using a text editor, create a document that contains the following ASP page code. This code creates the ADO connection and opens the proper **RecordSet**.

```
<%
   On Error Resume Next
'Create the ADO connection
   Set adoConnection = Server.CreateObject("ADODB.Connection")
'Open the ADO - OLE DB connection; here you must specify the name
'of the ATK OLE DB Provider and the name of the database
   Call adoConnection.Open( _
      "provider=ObjectStore Active Toolkit OLE DB Provider;" & _
      "data source=C:\odi\ATK6.0\Examples\demodbs\extrademo.db","","")
   If Err.Number<>0 Then  'Something went wrong
      Response.Write("Error: " & Err.Description)
   Else  'Connection has been correctly opened
'Create an ADO RecordSet
   Set adoRS = Server.CreateObject("ADODB.RecordSet")
'And open data view "tut3_table1" using the previously opened ADO connection
   Call adoRS.Open("allimages", adoConnection, 1)  '1 is adOpenKeySet
      If Err.Number<>0 Then  'Something went wrong
         Response.Write("Error: " & Err.Description)
      Else 'RecordSet has been correctly opened
'Generate the HTML output for the opened RecordSet
```

```
            WriteObjectManagerTable(adoRS)
        End If
    End If
%>
```

The **WriteObjectManagerTable** procedure scans the **RecordSet** and builds the HTML table that contains the images:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub WriteObjectManagerTable(anADORS)
    On Error Resume Next
'Remove any previous navigation information
    Session("ObjectManagers")=Empty
'Create an array that will store the array of bytes
'representing the images
    Dim  ObjectManagers()
    ReDim  ObjectManagers(anADORS.RecordCount)
'Initialize the HTML table
    Response.Write("<TABLE BORDER=1>")
'Generate the column heading
    Response.Write("<TH><B> --- Images --- </B></TH>")
'Scan all the records in the ADO RecordSet
'Keep track of the row number
    rowNumber=0
    anADORS.MoveFirst
    Do While Not anADORS.EOF
    Response.Write("<TR>")
'Retrieve the array of bytes that is the content of the image
'Object Manager; store it in the array in the proper position
    ObjectManagers(rowNumber)=anADORS(0).GetChunk(anADORS(0).actualsize)
'Create the link to the second page that actually displays the image
    Response.Write("<TD><CENTER><IMG SRC=""tut3_om2.asp?omPosition=" & _
        rowNumber & """></CENTER></TD></TR>")
'Move to the next row in the ADO RecordSet
    anADORS.MoveNext
    rowNumber=rowNumber+1
    Loop
    Response.Write("</TABLE>")
'Store the array of images in the Session
    Session("ObjectManagers")=ObjectManagers
End Sub
</SCRIPT>
```

2 Save this code in the **InetPub\ASPSamp\ATK\Tutorial3** directory as **tut3_om1.asp**.

2 Create another ASP page to display the data.

1 Create another ASP page to write the image data to the table.

2 Remove the default lines created by Visual InterDev. The output of the Active Server Page must match the type and the actual data that the multimedia Object Manager returns.

```
<%@ LANGUAGE="VBSCRIPT" %>
<%
'Set the content type of this block; we know they are jpeg images
    Response.ContentType="image/jpeg"
'Retrieve the index in the array where the image is
    omIndex=Request.QueryString("omPosition")
'Retrieve the array of images
    ObjectManagers=Session("ObjectManagers")
'Write the image bits
    Response.BinaryWrite(ObjectManagers(omIndex))
%>
```

3 Save this code in the **InetPub\ASPSamp\ATK\Tutorial3** directory as **tut3_om2.asp**.

3 Test the application.

To test the application, browse the **tut3_om1.asp** page. Open **http://localhost/ASPSamp/ATK/Tutorial3/tut3_om1.asp**:

Summary

In this chapter, you learned how to access ObjectStore data views through ADO, and how to customize the default instance formats of the data views using simple SQL commands.

You also wrote ASP code that can navigate nested **RecordSet** objects in ADO.

Finally, you wrote a sample ASP / ADO application that accesses multimedia data stored in an ObjectStore database and displays it as multimedia Object Manager instances.

Chapter 4
Creating an ODBC Data Source Using ATK

Introduction

Although OLE DB is a key component of Microsoft Corporation's Universal Data Access strategy, many applications still use ODBC as the protocol to access data sources. Therefore, it is important to be able to access an OLE DB provider from any ODBC consumer.

Software resources

To complete the exercise in this chapter, you need these software resources:

Resource	*Where to Find It*
Database	**odi\\ATK6.0\Examples\demodbs\carsdemo.db**
ISG Navigator	ISG International Software Group at **http://www.isgsoft.com** or Microsoft Corporation at **http://www.microsoft.com/data/**
Microsoft Access	Microsoft Corporation at **http://www.microsoft.com/access/**

In this chapter

In this chapter, you access the ObjectStore Active Toolkit OLE DB provider from an ODBC-compliant reporting tool. This chapter contains the following exercise:

Exercise	*Description*
Use ATK as an ODBC Data Source on page 70	Configure access to an ObjectStore database as an ODBC data source, and access it through an ODBC consumer.

Use ATK as an ODBC Data Source

Overview

To use ATK as an ODBC data source, you first access the ObjectStore Active Toolkit OLE DB provider. Next, create and access an ODBC data source.

Process

To create this application, follow these steps:

1 Configure ISG Navigator.

2 Create the ODBC data source.

3 Access the ODBC data source.

1 Configure ISG Navigator.

Configure ISG Navigator so it can access the ObjectStore Active Toolkit OLE DB provider:

1 Using a text editor, open **\ISGNav\Def\nav.bnd**.

2 In the **[TDP-NAMES]** section, add this line:

CARSDEMO = OLEFS

where **CARSDEMO** is a name that references the ATK Table Data Provider (TDP) and **OLEFS** is the type of data provider.

3 Add a section called **[CARSDEMO]** containing this line:

```
TDP_CONNECT =
ObjectStore Active Toolkit OLE DB Provider;c:\odi\ATK6.0\Examples\demodbs\carsdemo.db
```

where **ObjectStore Active Toolkit OLE DB Provider** is the name of the ObjectStore Active Toolkit OLE DB provider and **c:\odi\ATK6.0\Examples\demodbs\carsdemo.db** is the full pathname of the ObjectStore database you want to access though an ODBC consumer.

The modified file looks like this:

```
[TDP-NAMES]
CARSDEMO = OLEFS
[CARSDEMO]
TDP_CONNECT =
ObjectStore Active Toolkit OLE DB provider;c:\odi\ATK6.0\Examples\demodbs\carsdemo.db
```

4 Save the **nav.bnd** file.

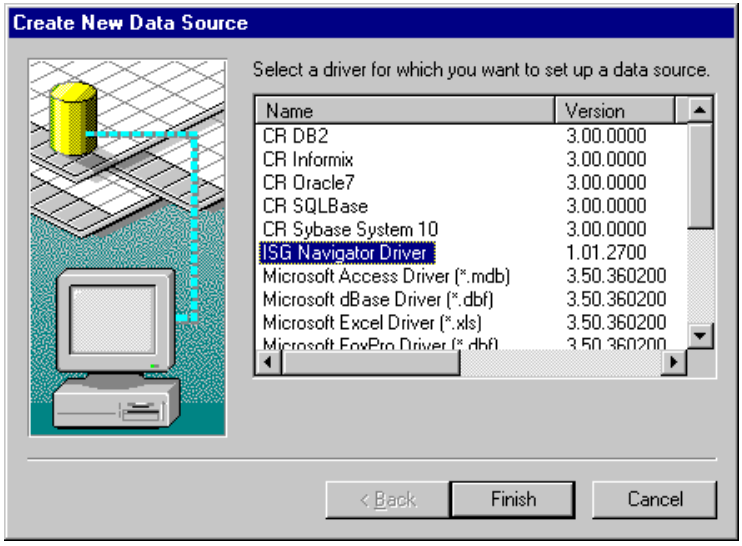
2 Create the ODBC data source.

Create a new ODBC data source that points to the **carsdemo.db** database through the ISG Navigator Driver / ObjectStore Active Toolkit OLE DB provider.

1 Open the Windows Control Panel and run the ODBC 32 configuration program.

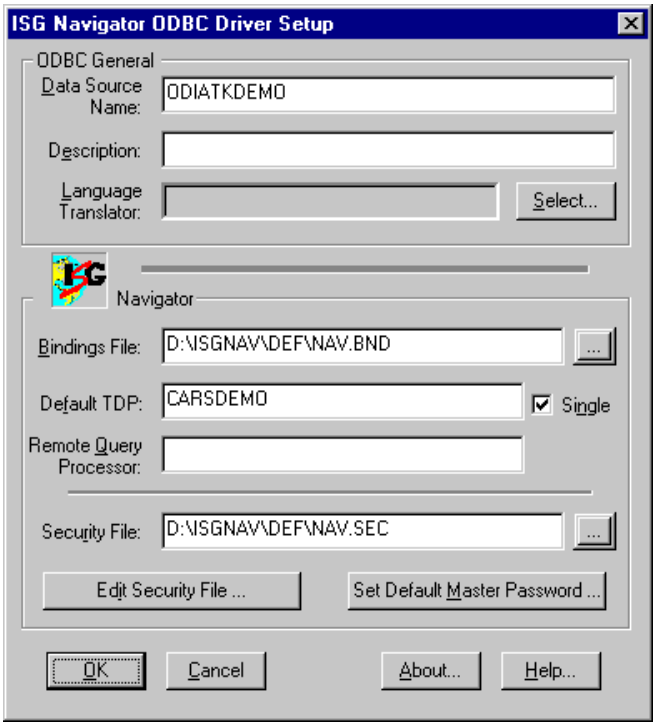
2 Click **Add** to create a new ODBC source.

3 From the list of available drivers, select **ISG Navigator Driver**.



4 Enter **ODIATKDEMO** as the **Data Source Name**.

5 Enter **CARSDEMO** as the **Default TDP Name**. This is the name you specified in the **[TDP-NAMES]** section of the **nav.bnd** configuration file.



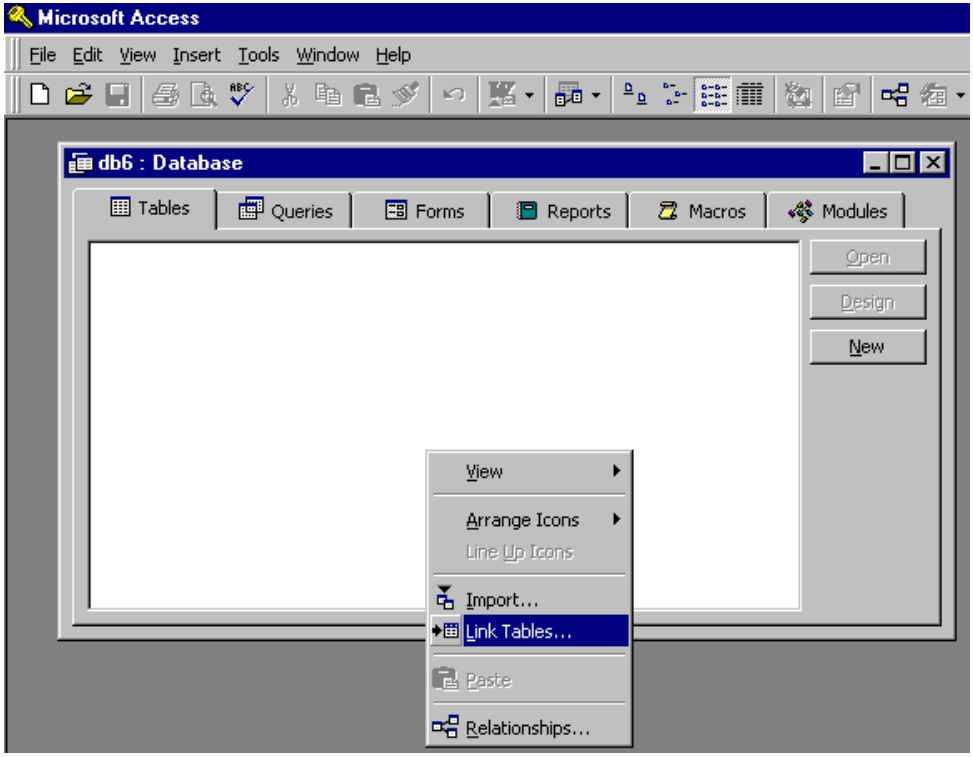
3 Access the ODBC data source.

Open the ATK-linked ODBC source using an ODBC consumer application, such as Microsoft Access.

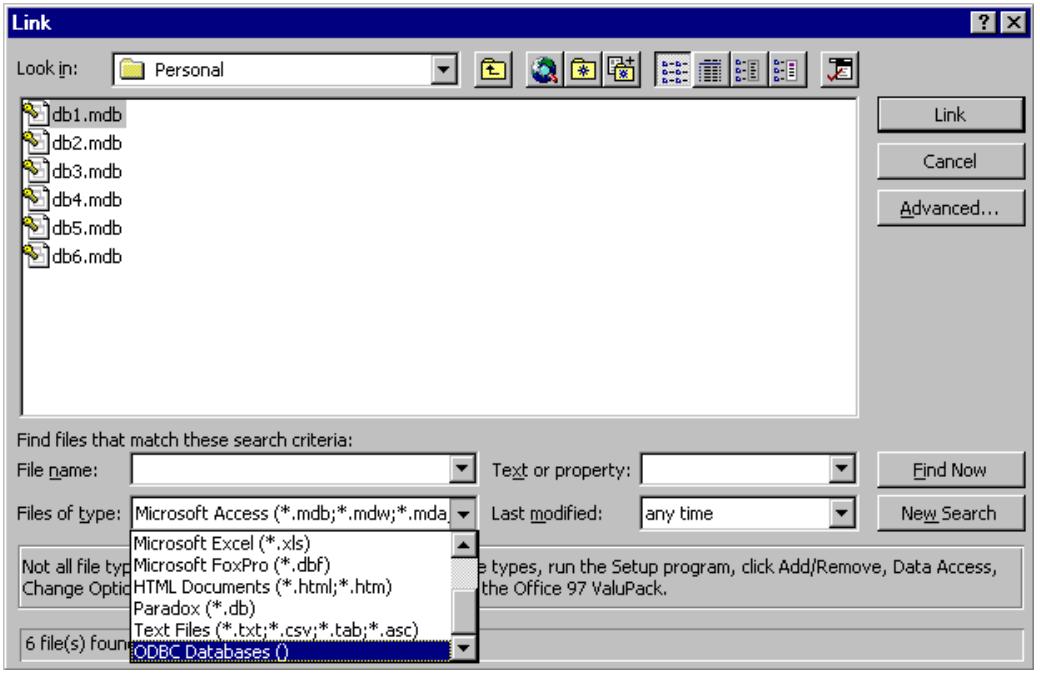
1 Start Access.

2 Create a new database.

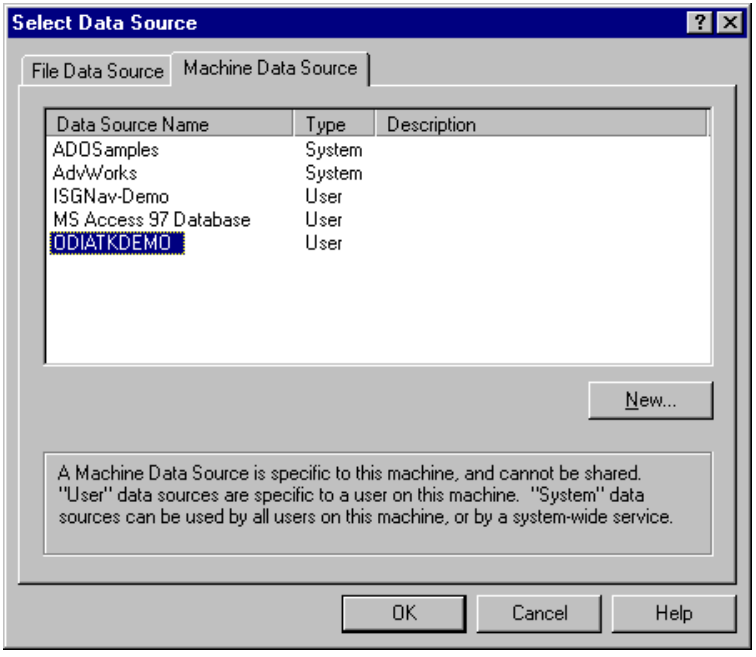
3 Select **Link Tables** from the **Tables** sheet shortcut menu; this will enable you to insert a linked table into the database.



4 Select the **ODBC Databases** file type.

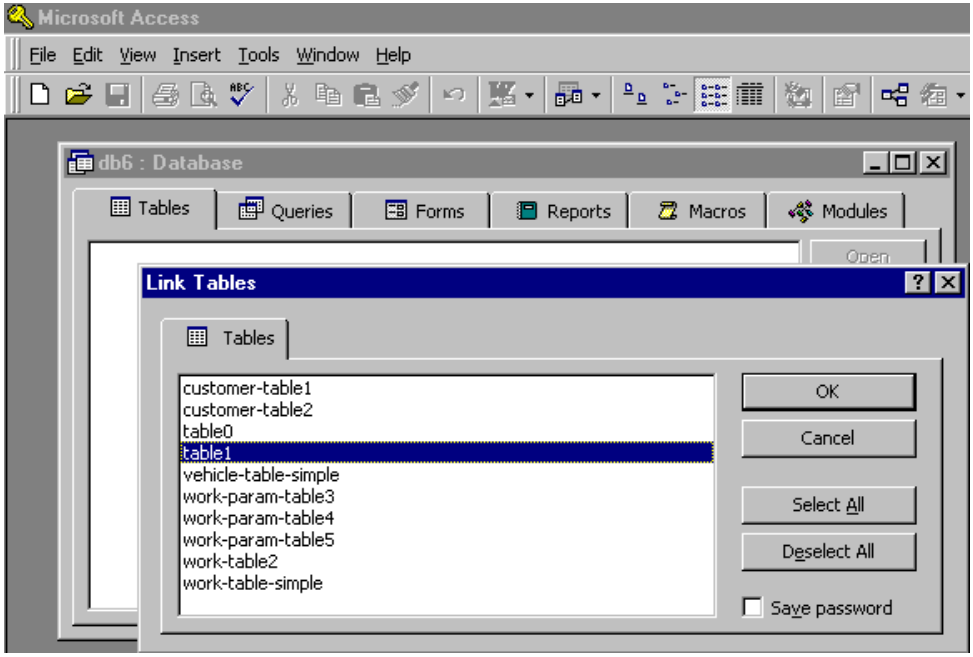


5 On the **MachineData Source** sheet of the Select Data Source dialog box, choose **ODIATKDEMO** and click **OK**.



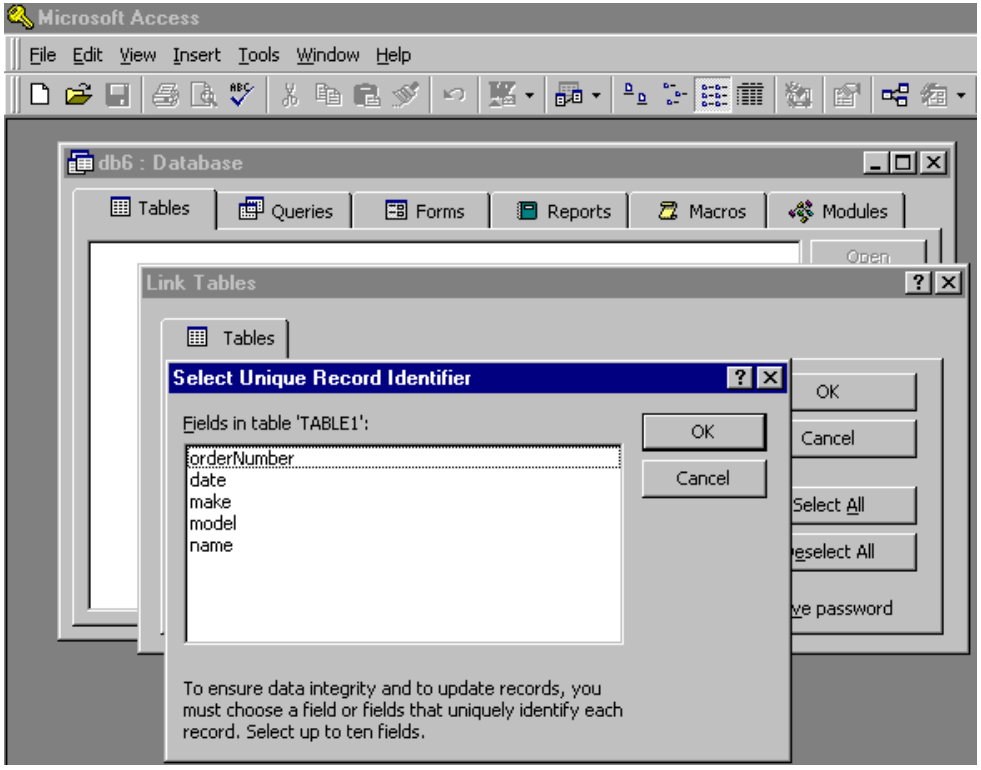
The Link Tables dialog box appears. It lists all the data views defined in the **carsdemo.db**.

6 Select **table1**.



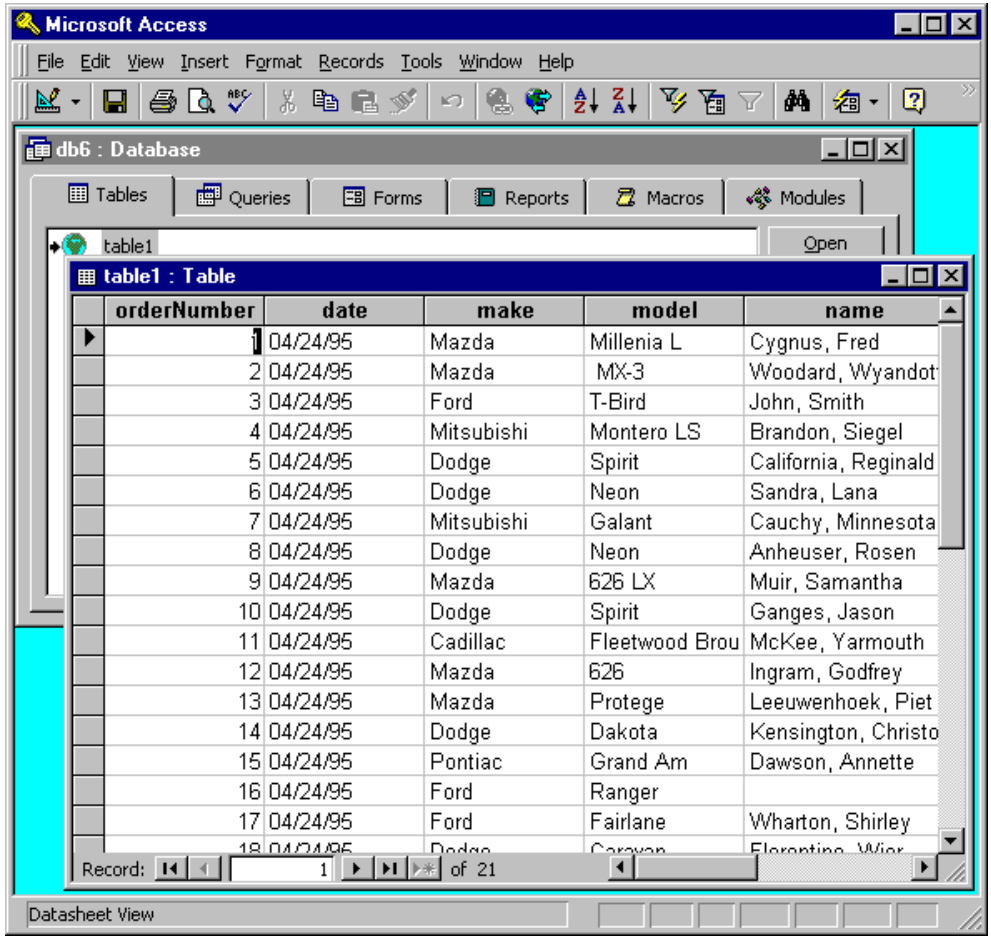
Access inspects the **table1** data view and retrieves the names and definitions of its columns.

7 Click **OK** without specifying an identifier field.



Now you can browse **table1** using Access.

8 Double-click on **table1** to list its records.


Microsoft Access
File Edit View Insert Format Records Tools Window Help
db6 : Database
Tables Queries Forms Reports Macros Modules
table1
Open
table1 : Table
orderNumber date make model name
1 04/24/95 Mazda Millenia L Cygnus, Fred
2 04/24/95 Mazda MX-3 Woodard, Wyandot
3 04/24/95 Ford T-Bird John, Smith
4 04/24/95 Mitsubishi Montero LS Brandon, Siegel
5 04/24/95 Dodge Spirit California, Reginald
6 04/24/95 Dodge Neon Sandra, Lana
7 04/24/95 Mitsubishi Galant Cauchy, Minnesota
8 04/24/95 Dodge Neon Anheuser, Rosen
9 04/24/95 Mazda 626 LX Muir, Samantha
10 04/24/95 Dodge Spirit Ganges, Jason
11 04/24/95 Cadillac Fleetwood Brou McKee, Yarmouth
12 04/24/95 Mazda 626 Ingram, Godfrey
13 04/24/95 Mazda Protege Leeuwenhoek, Piet
14 04/24/95 Dodge Dakota Kensington, Christo
15 04/24/95 Pontiac Grand Am Dawson, Annette
16 04/24/95 Ford Ranger
17 04/24/95 Ford Fairlane Wharton, Shirley
Record: 1 of 21
Datasheet View

Summary

In this chapter, you configured ISG Navigator so that you could access the ObjectStore Active Toolkit OLE DB provider. Next, you created an ODBC data source based on an ObjectStore database. Then you accessed the data source using an ODBC consumer, which displayed the data returned by its query.

Once you have configured ISG Navigator to work with ATK, you can access the ATK OLE DB data source from any ODBC-compliant tool, such as Microsoft Access, Crystal Reports, or any other ODBC consumer.

Chapter 5
Using Crystal Reports with ATK

Introduction

Reporting functionality is a typical application requirement, but most commonly used reporting tools are designed to run against relational databases. ATK, however, lets you easily build tabular views of the data in an ObjectStore database using most common report generating tools, including Crystal Reports.

Crystal Reports, and other ODBC consumer applications, can access ATK's OLE DB interface through ISG Navigator. This capability enables you to add reporting functionality to your ObjectStore application.

Software resources

To complete this tutorial, you need the following software resources:

Resource	*Where to Find It*
Database	**\odi\ATK6.0\Examples\demodbs\carsdemo.db**
Crystal Reports Version 4.x or later	Seagate Software at **http://www.seagate.com/**

In this chapter

In this chapter, you create a data view that you can use with any reporting tool. Then, you use Crystal Reports to create a report. This chapter contains these sample exercises:

Exercise	*Description*
Create a Data View for the Report on page 81	Create and customize a data view that provides access to specific data items.

Exercise	*Description*
Create a Report Using Crystal Reports on page 83	Using the Crystal Reports wizard, create a report that displays data from the fields in the data view.

Create a Data View for the Report

Process

Specify the data that is available for the report. To do so, follow these steps:

1 Create a new data view.

2 Customize the data view.

3 Save the data view.

1 Create a new data view.

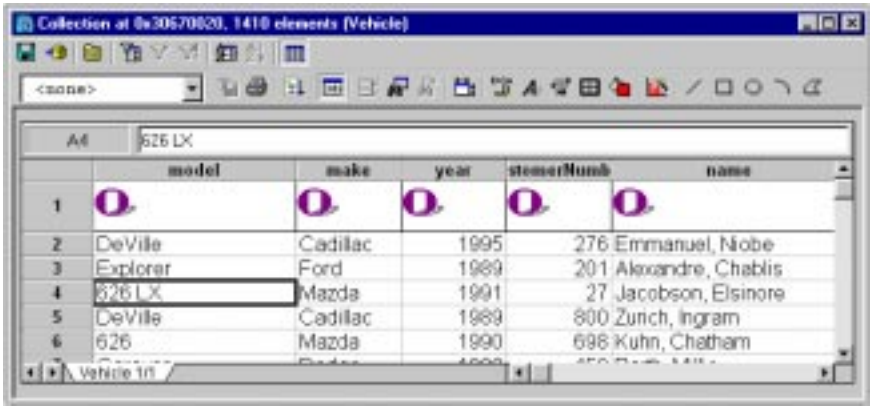
1 Start Inspector.

2 Open the **carsdemo.db** database.

3 Double-click on the **vehicle** root in the Database Roots pane.

4 Select **Data View | Create**.

This data view displays a collection of vehicles:



2 Customize the data view.

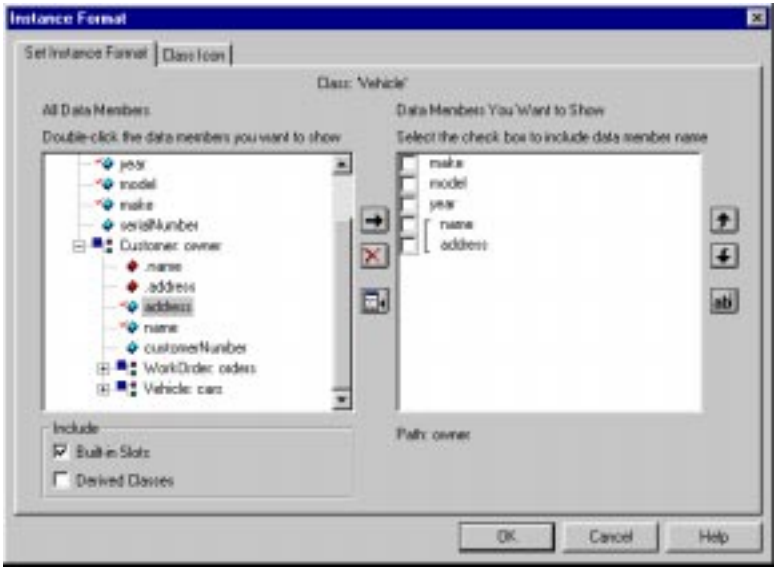
Modify the **Vehicle** class instance format to display owner information for each vehicle.

1 Right-click anywhere in the data view.

2 Select **Set Format of Class**.

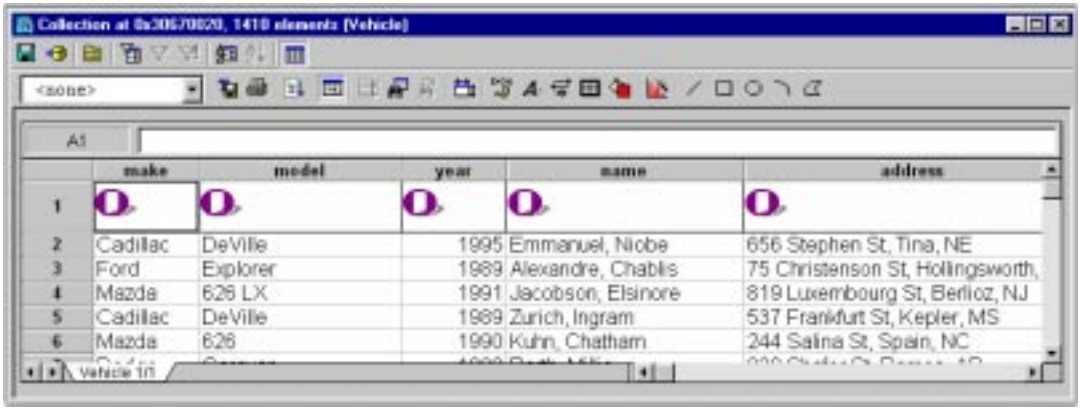
3 Expand the **owner** relationship.

4 On the **Instance Format** sheet of the **Instance Format** dialog box, double-click on the **name** and **address** data members in the **Customer** class.



5 Click **OK**.

For each vehicle, Inspector traverses pointers to display the owner name and address in the new instance format:



3 Save the data view.

1 Select **File | Save**.

2 Name the data view **VEHICLES**.

3 Select **File | Save All**.

Create a Report Using Crystal Reports

Overview

Using the ObjectStore Active Toolkit OLE DB provider and ISG Navigator, you can access an ObjectStore database from any ODBC consumer application.

Process

To do this, follow these steps:

1 Start the Report Wizard.

2 Create the report.

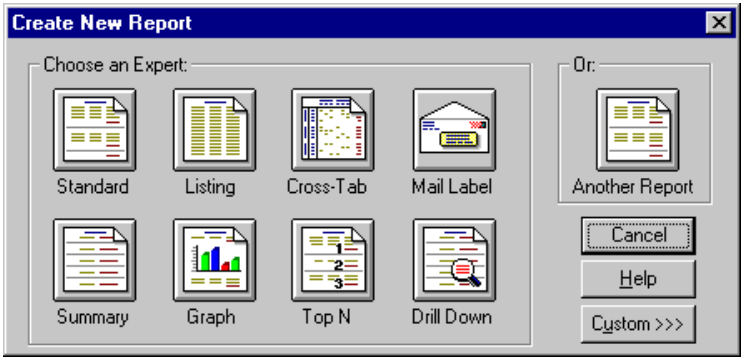
3 Group the report data by year.

4 Preview the report.

1 Start the Report Wizard.

1 Start Crystal Reports.

2 Select **File | New**.

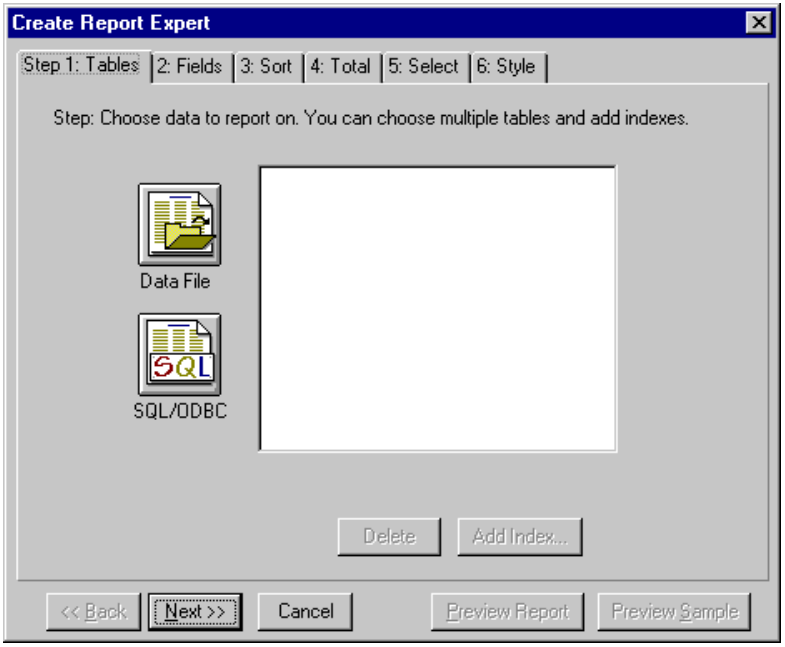


The Create New Report dialog box appears.

3 Click **Standard** to start the Crystal Reports wizard.

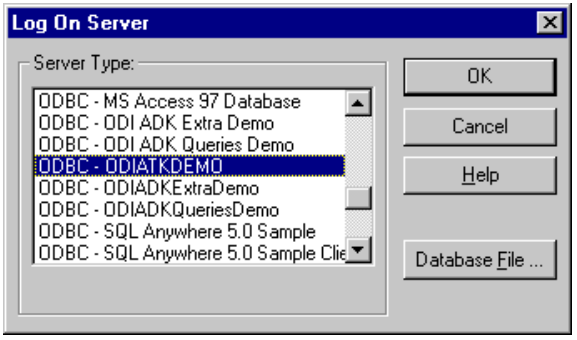
4 Click **SQL/ODBC**.

The Create Report Expert dialog box appears:

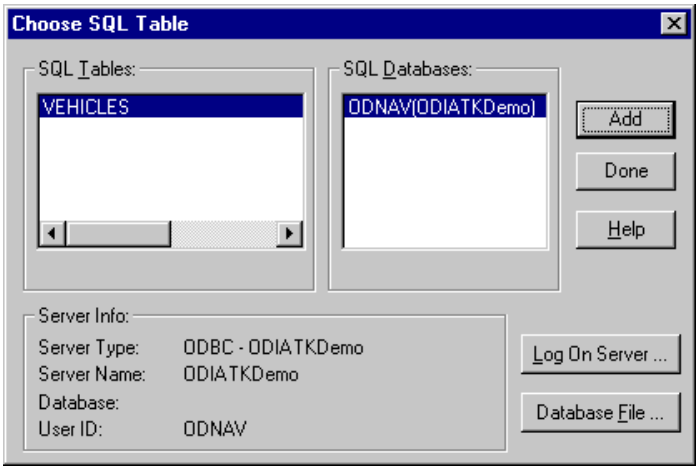


5 Click the **SQL/ODBC** tool.

6 From the list of available data sources, choose **ODBC-ODIATKDEMO**.



7 Select the **VEHICLES** data view, which you defined in Inspector.



If the data view is not listed, verify that Inspector is saving its metaknowledge in the same place from which ATK is loading it. Refer to Chapter 4, Configuring ATK, in the *ObjectStore Active Toolkit Reference*.

8 Click **Add**.

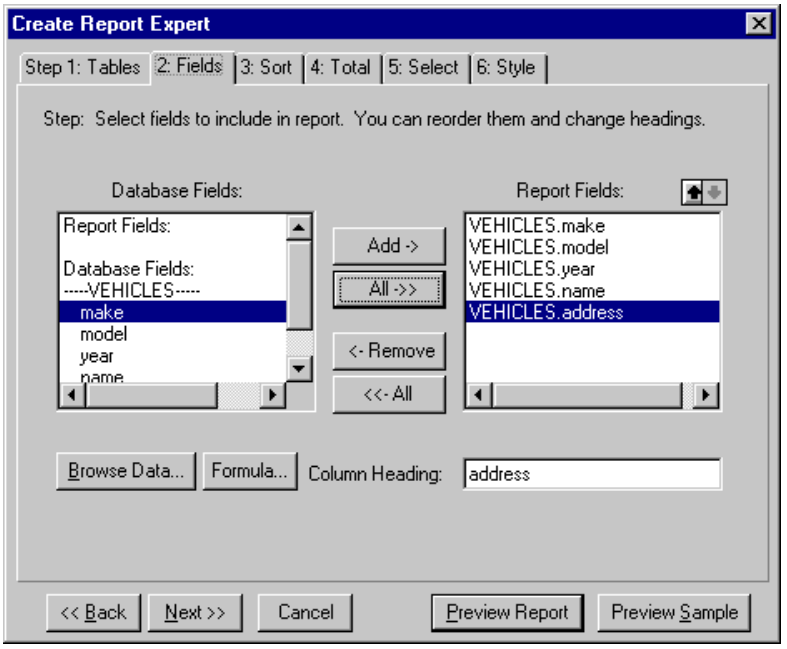
9 Click **Done**.

2 Create the report.

1 Follow the directions of the Crystal Report wizard to create the report.

2 On the Create Report Expert dialog box, click the **2: Fields** tab.

3 To include all the columns defined in the **VEHICLES** data view, click the **All->>** button.



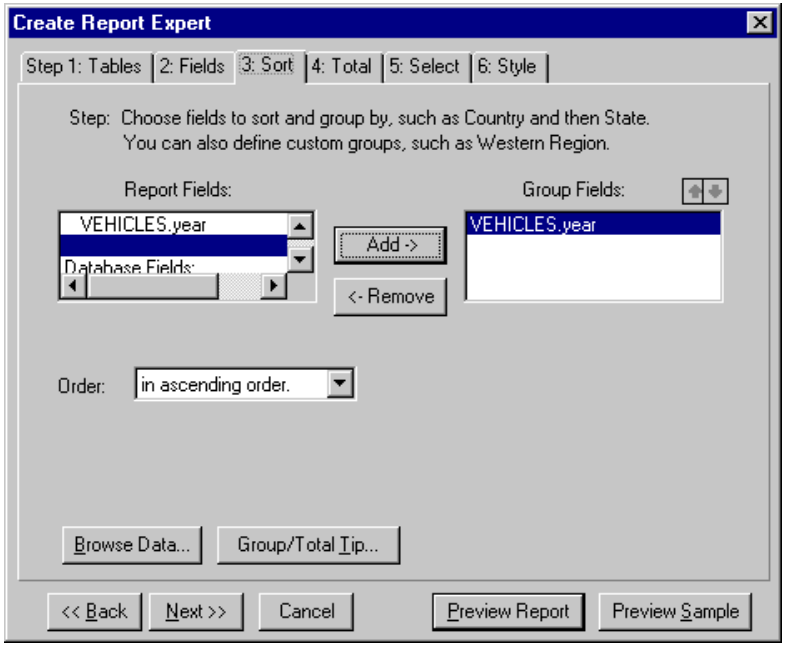
Including all columns of the **VEHICLES** data view requires that data members from two different classes be included.

3 Group the report data by year.

1 Click the **3: Sort** tab.

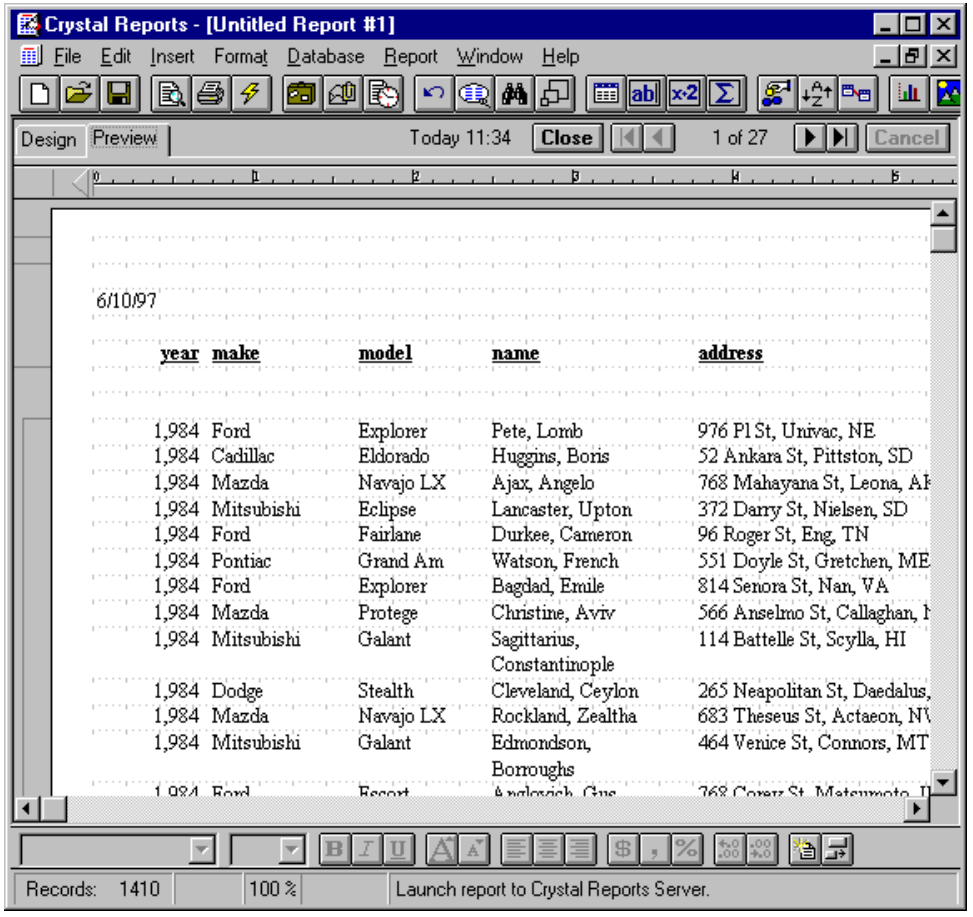
2 Select **VEHICLES.year** from the **Report Fields** pane.

3 Click **Add**. **VEHICLES.year** appears in the **Group Fields** pane.



4 Preview the report.

Preview the report by clicking the **Preview Report** button on the **3: Sort** sheet of the Create Report Expert dialog box.



Now that you have created the report, you can modify it using Crystal Reports features.

Summary

In this chapter, you created a data view that makes particular data items available for querying. Then you used the Crystal Reports wizard to create a report that queries an ODBC data source and displays the data items that you specify.

Generating reports based on data views that you build with Inspector and access through ATK is a straightforward process. By using data views, you can generate reports for any ObjectStore database.

Chapter 6
Using ATK ActiveX Server from DCOM

Introduction

If you are developing a distributed web application that uses ActiveX controls, you can use Distributed COM (DCOM) to access them and query a particular ATK ActiveX Server. For example, the ATK grid control has a **Server** property with which you can specify the ActiveX server host machine.

By working with a remote ATK ActiveX server, you can build applications that can display data from remote ObjectStore databases without direct access to an ObjectStore client. For example, you can use the ATK ActiveX grid control to display an ObjectStore data view even if no ObjectStore client or server is running on your local machine, and the ObjectStore database is not directly reachable from your machine.

Software requirements

To complete the exercises in this chapter, you need these software resources:

Resource	***Where to Find It***
Database	**\odi\ATK6.0\Examples\demodbs\carsdemo.db**
ActiveX grid control	**\odi\ATK6.0\bin\ATKCtrls.ocx**
Internet Explorer Version 4.0 or later	**http://www.microsoft.com**
Microsoft Front Page Version 2.0 or later	**http://www.microsoft.com**

In this chapter

In this chapter, you configure DCOM and test its remote connection between an ATK ActiveX grid control and an ATK ActiveX server. This chapter contains the following sample exercises:

Exercise	*Description*
Create an HTML Page Using the ATK Grid Control on page 93	Create an HTML page containing an ATK ActiveX grid control, and write a script that loads the control with data. Run the application locally.
Access the Page Remotely on page 98	Access the page from a remote client.

Create an HTML Page Using the ATK Grid Control

Overview

In this exercise, you create an HTML page that contains an ActiveX grid control. Then you write a script that queries the ATK ActiveX server and loads the ATK ActiveX grid control with data.

Process

To create this application, follow these steps:

1 Create an HTML page that contains an ActiveX control.

2 Write a script that loads the ATK grid control with data.

3 Test the application.

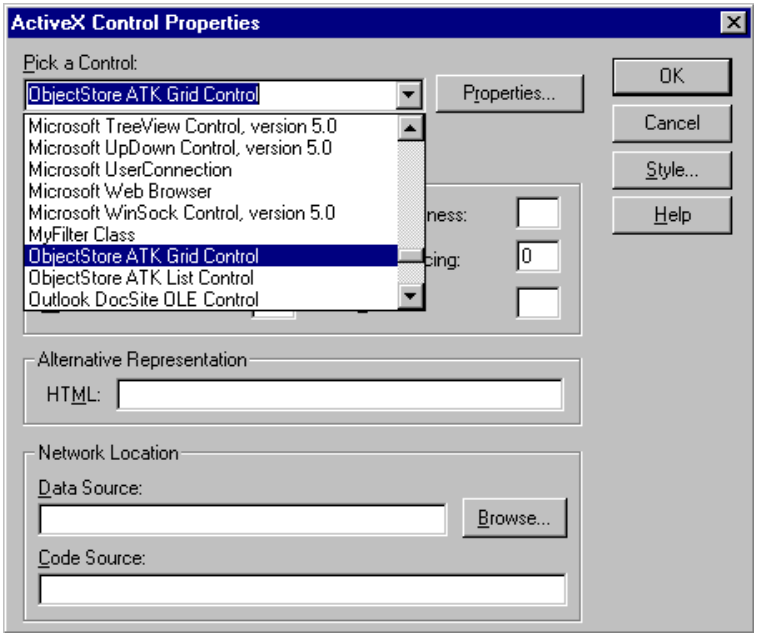
1 Create an HTML page that contains an ActiveX control.

1 Using FrontPage, create a new, blank HTML page. (This page is displayed in Microsoft Explorer.)

2 Insert an ActiveX control by clicking **Insert | Advanced | Active X Control** on the menu bar.

The ActiveX Control Properties dialog box appears.

3 Select the **ObjectStore ATK Grid Control** from the list of available ActiveX controls.



4 Set the control **Name** to **ATKGrid**, and edit the control properties.

5 Modify these three property values:

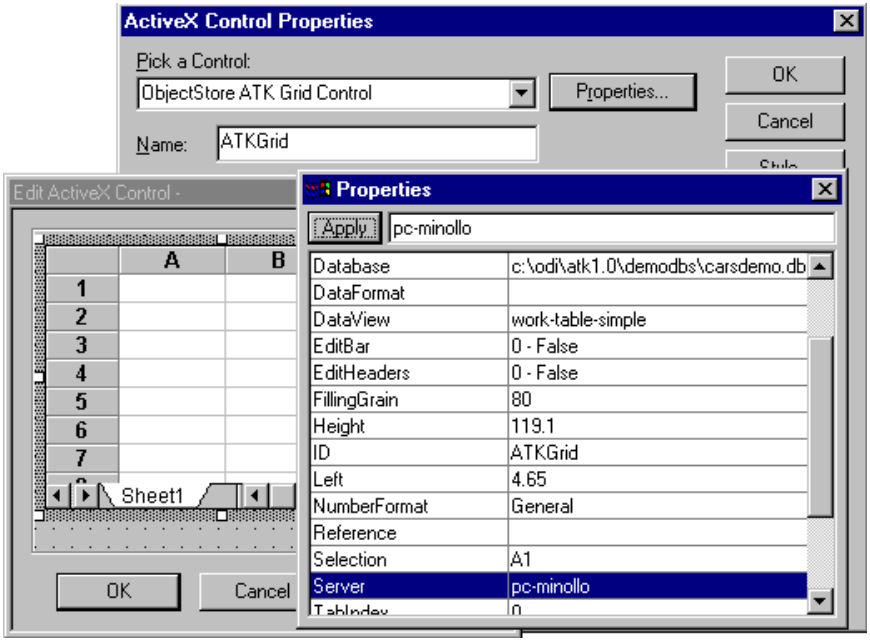
Database = c:\odi\ATK6.0\Examples\demodbs\carsdemo.db
DataView = work-table-simple
Server = PC-SERVER

where

Database is the complete path of the ObjectStore database you want to inspect, relative to the server machine.

DataView is the name of the data view you want to display inside the ATK grid control, and **work-table-simple** is the data view defined as an example in the **carsdemo** database shipped with ATK.

Server is the name of the machine where the ATK ActiveX kernel is running, such as **PC-SERVER**.



6 Click **OK** to create the control and close the **Properties** dialog box.

2 Write a script that loads the ATK grid control with data.

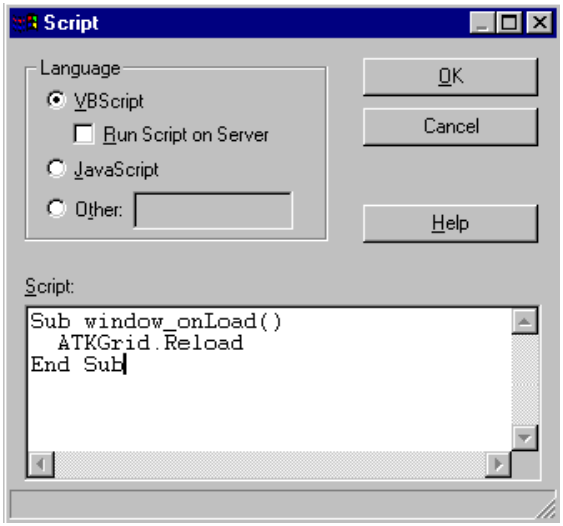
Write a small script that instructs the ATK grid control to load its content whenever the HTML page is loaded.

1 In FrontPage, select **Insert | Advanced | Script** from the menu bar.

2 In the Script box, enter this code:

```
Sub window_onLoad()
    ATKGrid Reload
```

End Sub



ATKGrid exposes the **Reload** method, which connects the ActiveX control to the specified server and fills the control's cells with the contents of the specified data view. Internet Explorer executes the **window_onLoad()** procedure automatically whenever the HTML page is loaded.

3 Select **OK**.

4 Save the new page in the **InetPub\ASPSamp\ATK\Tutorial6** directory as **tut_ocx1.htm**.

3 Test the application.

If you stored this page on the **PC-SERVER** machine (the same specified in the **Server** property of the ATK grid control), you can open it from a server running on **PC-SERVER**. Because the page and server are on the same machine (that is, everything runs locally), there is no need to configure DCOM.

1 Open the URL **http://localhost/ASPSamp/ATK/Tutorial6/tut_ocx1.htm**.

	derNumb	date	name	address
1	1	04/24/95	Cygnus, Fred	842 Goodrich St, Ron, SC
2	2	04/24/95	Woodard, Wyandotte	555 Waterman St, Christina
3	3	04/24/95	John, Smith	366 Carlyle St, Maurice, AZ
4	4	04/24/95	Brandon, Siegel	545 Donna St, Doubleday, N
5	5	04/24/95	California, Reginald	282 Freedman St, Christens
6	6	04/24/95	Sandra, Lana	615 Collins St, Matson, IN
7	7	04/24/95	Cauchy, Minnesota	422 Sabina St, Kowalski, MI
8	8	04/24/95	Anheuser, Rosen	720 Corinthian St, Russia, N
9	9	04/24/95	Muir, Samantha	840 Ramo St, Algonquin, NE
10	10	04/24/95	Ganges, Jason	801 Varitype St, Ryan, SD
11	11	04/24/95	McKee, Yarmouth	957 Atlantic St, Gregg, ID
12	12	04/24/95	Ingram, Godfrey	387 Dalton St, Rd, MD
13	13	04/24/95	Leeuwenhoek, Piet	123 Knauer St, Faber, AL
14	14	04/24/95	Kensington, Christopher	52 Waterbury St, Philip, AZ
15	15	04/24/95	Dawson, Annette	489 Douglass St, Bruce, WI
16	16	04/24/95		
17	17	04/24/95	Wharton, Shirley	647 Gothic St, Nadine, KS
18	18	04/24/95	Florentine, Wier	921 Weinberg St, Siegel, IA

Access the Page Remotely

Overview

You can access the same HTML page that contains an ActiveX control and resides on a server, such as **PC-SERVER**, remotely from a client machine, such as **PC-CLIENT**. This exercise shows you one way to do this.

Prerequisite

In order to access an HTML page from a client machine, the client must have the ATK ActiveX controls installed. This option is available in the ATK installation.

Process

To create this application, follow these steps:

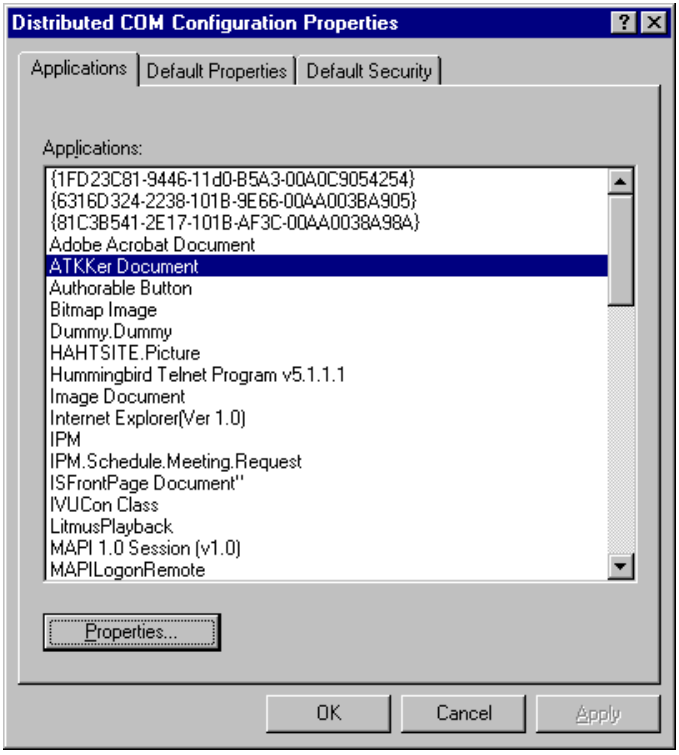
1 Check the DCOM configuration of the server.

2 Test the application.

1 Check the DCOM configuration of the server.

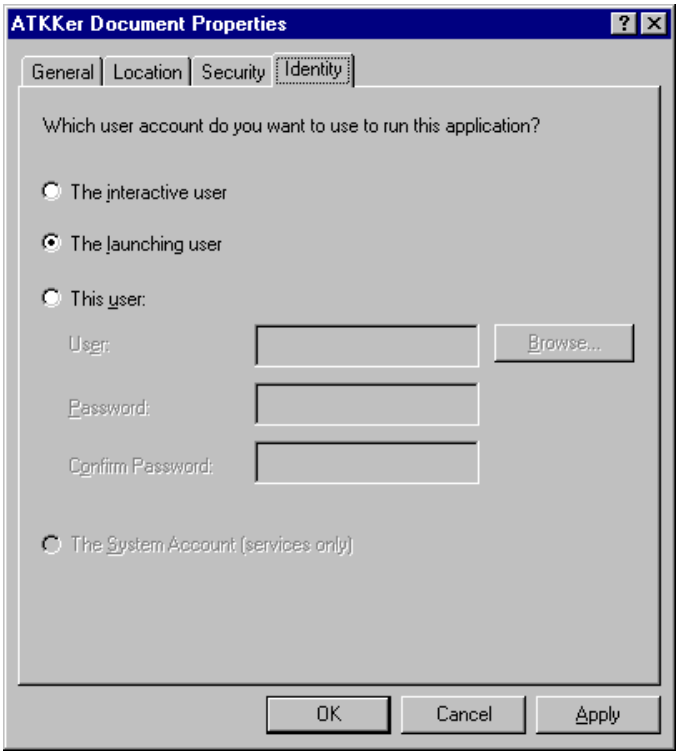
1 From the **Start** menu, run the Windows DCOM configuration utility, **dcomcnfg**.

The **Distributed COM Configuration Properties** dialog box appears.



2 On the Applications sheet, select the **ATKKer.Document** application (that is, the ID that identifies the ATK ActiveX server), and click the **Properties** button to view and edit the DCOM settings of the ATK ActiveX server.

The **ATKKer Document Properties** dialog box appears.



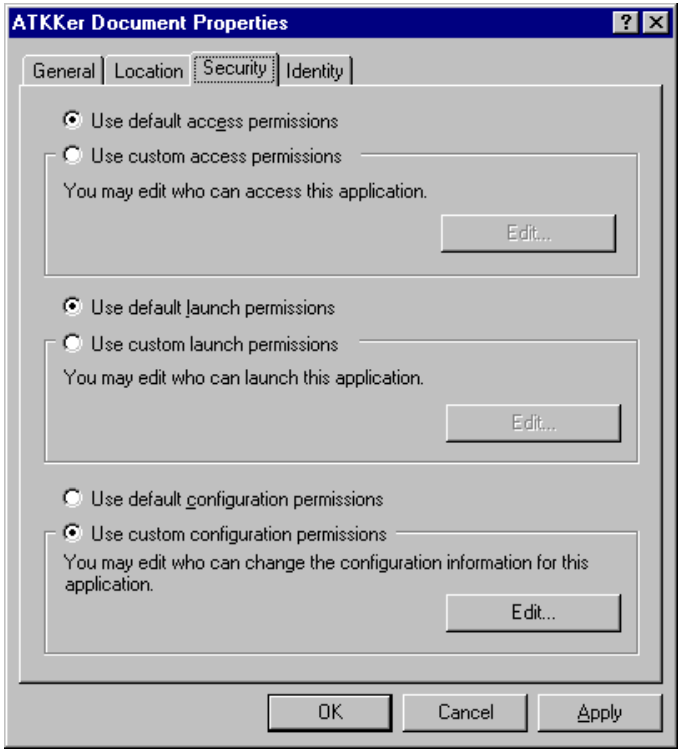
3 Click the **Identity** tab and specify the user account that remotely runs the ATK ActiveX server.

The default is the **launching user**, which means that the user who is running the remote client must also have an account on the server machine. Use this setting when you are developing an Intranet application, and there is a Windows NT server that handles all the user accounts.

You can also enter a specific client user account, or make ATK ActiveX server run as the user who is currently logged on to the server.

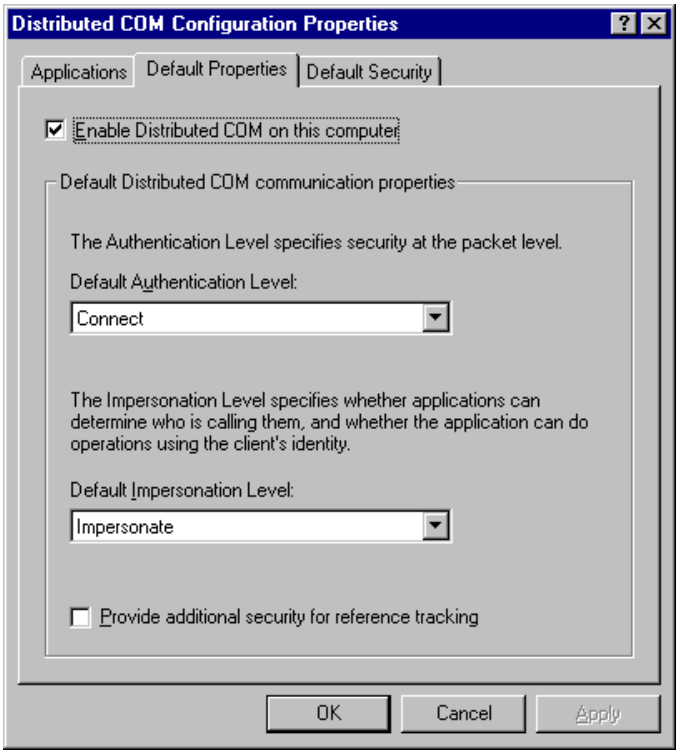
In this exercise, run the ATK ActiveX server using a launching user account.

4 Click the **Security** tab and customize the access, launch, and configuration permissions of the ATK ActiveX server.



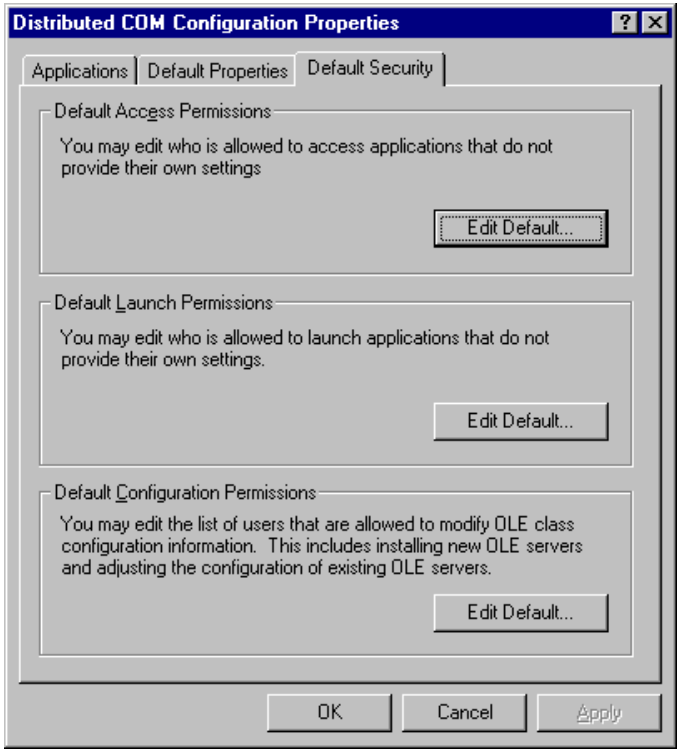
5 Click **OK** to use the default settings and to return to the Distributed COM Configuration Properties dialog box.

6 Click the **Default Properties** tab and set the default properties for DCOM on the server.



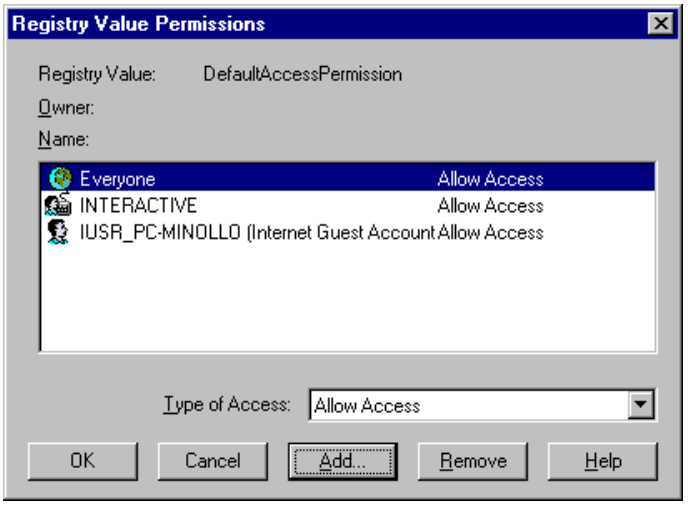
7 Select the **Connect** authentication level and the **Impersonate** impersonation level to provide acceptable security, and click on **OK**.

8 Click the **Default Security** tab and specify which users can remotely access and launch the ActiveX servers on your server machine.



9 Verify that the **Everyone** user account is included in both the **Default Access Permissions** and **Default Launch Permissions**:

Click the **Edit Default** button to display the Registry Value Permissions dialog box



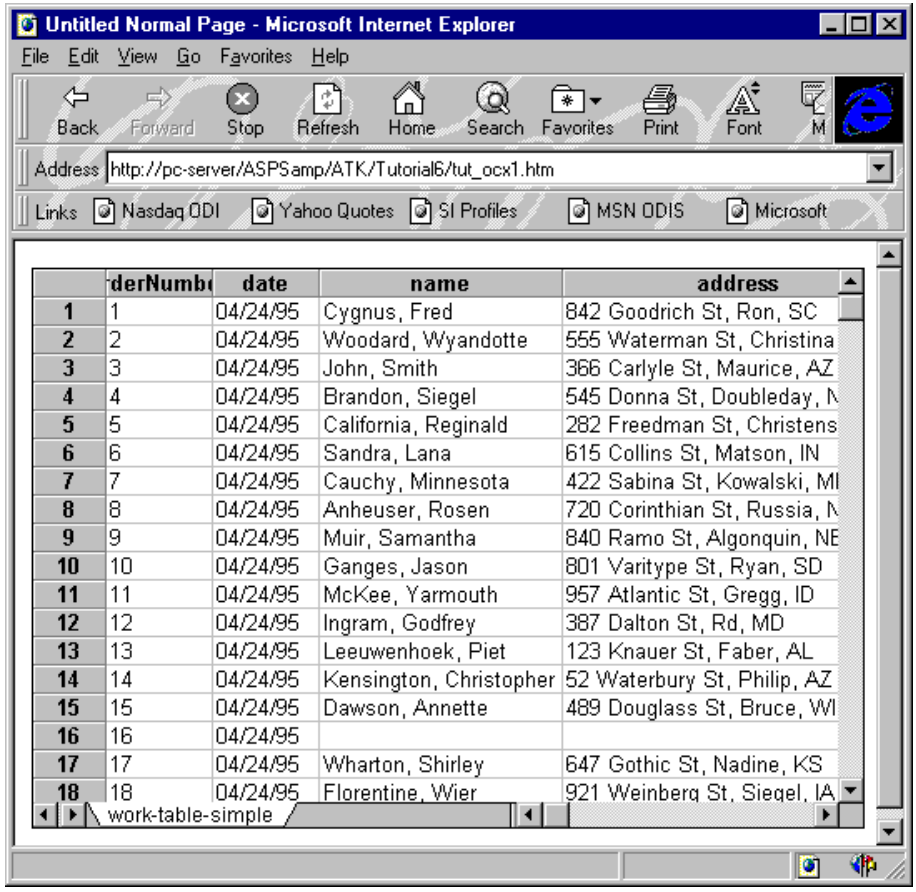
10 Click **OK** to exit from the DCOM configuration utility on the **PC-SERVER**.

2 Test the application.

Connect to the **PC-SERVER** ATK ActiveX server from the **PC-CLIENT** workstation.

Open the URL **http://pc-server/ASPSamp/ATK/Tutorial6/tut_ocx1.htm**.

Remote access provides the same result as local access:



	derNumb	date	name	address
1	1	04/24/95	Cygnus, Fred	842 Goodrich St, Ron, SC
2	2	04/24/95	Woodard, Wyandotte	555 Waterman St, Christina
3	3	04/24/95	John, Smith	366 Carlyle St, Maurice, AZ
4	4	04/24/95	Brandon, Siegel	545 Donna St, Doubleday, N
5	5	04/24/95	California, Reginald	282 Freedman St, Christens
6	6	04/24/95	Sandra, Lana	615 Collins St, Matson, IN
7	7	04/24/95	Cauchy, Minnesota	422 Sabina St, Kowalski, MI
8	8	04/24/95	Anheuser, Rosen	720 Corinthian St, Russia, N
9	9	04/24/95	Muir, Samantha	840 Ramo St, Algonquin, NE
10	10	04/24/95	Ganges, Jason	801 Varitype St, Ryan, SD
11	11	04/24/95	McKee, Yarmouth	957 Atlantic St, Gregg, ID
12	12	04/24/95	Ingram, Godfrey	387 Dalton St, Rd, MD
13	13	04/24/95	Leeuwenhoek, Piet	123 Knauer St, Faber, AL
14	14	04/24/95	Kensington, Christopher	52 Waterbury St, Philip, AZ
15	15	04/24/95	Dawson, Annette	489 Douglass St, Bruce, WI
16	16	04/24/95		
17	17	04/24/95	Wharton, Shirley	647 Gothic St, Nadine, KS
18	18	04/24/95	Florentine, Wier	921 Weinberg St, Siegel, IA

Summary

In this chapter, you built an HTML page containing an ATK ActiveX control, and set up your server machine to make the ATK ActiveX server remotely accessible by means of DCOM.

The source code for the HTML page described in this chapter is located in **\ATK\Examples\Tutorial6\tut_ocx1.htm**.