



## ActXDoc (ActXDoc.vbp)

The ActXDoc is a simple ActiveX document-based application. Two ActiveX documents (FirstDoc.vbd, and SecndDoc.vbd) incorporate features such as passing data from one document to another, as well as use of the **HyperLink** object. See Chapter 5, "Creating an ActiveX Document," in "Creating ActiveX Components" of the *Component Tools Guide*.

<b>File</b>	<b>Description</b>
FrmAbout.frx	Binary data for the frmAbout.frm.
FrmAbout.frm	An About Box for the ActiveX document.
FrmAux.frm	An auxiliary form shown from the FirstDoc ActiveX document.
mGlobal.bas	Code module that contains global constants.
Actxdoc.vbp	The project file for the application.
FirstDoc.dob	The UserDocument object for the FirstDoc ActiveX document.
Secnddoc.dob	The UserDocument for the SecndDoc ActiveX document.

### To Run

From the Visual Basic **File** menu, click **Open Project** and select the ActXDoc.Vbp file, which is listed in the \Samples\CompTool\ActXDoc subdirectory of the main Visual Basic directory. Press F5 to run the application.

Run Internet Explorer 3.0 or later, click **Address**, and type the path of the FirstDoc.vbd file. The file will be found in the root directory of Visual Basic. For example, if you have installed Visual Basic in the default directory, this path will be "c:\Program Files\MSDevStudio\VB\FirstDoc.vbd")

## ATM (ATM.vbp)

The ATM.vbp sample application demonstrates how to use a resource file. This application produces the screen for an automated teller machine (ATM) that lets the user perform a bank transaction in one of several languages, such as English, French, or German.

### Background

The ATM.vbp sample prompts the user to specify the language to conduct the transaction in. Depending on the user's choice, the sample loads the appropriate group of resources from the ATM.res file to continue the transaction. For more information, see Chapter 4, "Managing Projects," in the *Programmer's Guide*.

File	Description
Atm.bas	Module containing procedures for the sample.
Atm.rc	Resource source file containing all localized strings and references to ICO and WAV files, which are built into ATM.RES.
Atm.res	Resource file containing text, icons, and sound (.WAV) data. Results from running a resource compiler on the ATM.RC file.
ATM.vbp	The project file for this sample.
ATM32.res	File containing resources for 32-bit systems.
FrmAmoun.frm	Form showing conversion of currencies into dollars.
FrmInput.frm	Form for input of user information
Openbank.frm	Form for selecting a language.
Openbank.frx	Binary data for the OPENBANK.FRM.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the ATM.vbp file, which is listed in the \Samples\Pguide\ATM subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## Biblio (Biblio.vbp)

This sample allows you to browse the Biblio.mdb database, which is located in the main Visual Basic directory (\VB). The sample demonstrates the **Data** control. For more information, see Chapter 12, "Accessing Databases with the Data control," in the *Programmer's Guide*.

<b>File</b>	<b>Description</b>
Biblio.bas	Module containing Public Sub and Function procedures.
Biblio.frm	The main form for the sample application.
Biblio.hlp	Help file for the sample.
Biblio.vbp	Project file for the sample.
Frmabout.frm	Form for the <b>About</b> box.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Biblio.vbp file, which is listed in the \Samples\PGuide\Biblio subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## Blanker (Blanker.vbp)

This sample demonstrates some of the graphics techniques found in Chapter 15, "Creating Graphics for Applications," of the *Programmer's Guide*. The sample uses the **Image**, **Line**, **PictureBox**, **Shape**, and **Timer** controls.

### Background

The sample demonstrates several Visual Basic graphic methods, including: **Circle**, **Line**, **Move**, and **PSet**.

File	Description
Blanker.frm	The main form for the sample.
Blanker.frx	Binary data for the Blanker.frm file.
Blanker.vbp	The project file for the sample.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Blanker.vbp file, which is listed in the \Samples\PGuide\Blanker subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## BookSale (Booksale.vbg)

This project demonstrates the use of an Automation server to encapsulate the logic of business policies and rules and to provide black box services to an external User Interface component. The client project is dedicated to delivering a clear and intuitive user interface for the user to select control options and view processing results. The client project cares a lot about how the user works and how they use the applications results, but it knows nothing about the business or operational rules of the application. The server component is dedicated to encapsulating business and data access rules into "sanctioned" services that client components use to find the information they need. The server has no idea how the user options are selected or how the results are presented to the user. This lack of specific user knowledge helps keep the server's functionality general, and as a result should increase its reusability potential for other applications. The server also uses class modules to structure the logic of its business and data access rules in a manner that aids development, debugging, readability, maintainability, and source code reusability.

The component can be run on the same machine as the client component, (which can be a significant aid in the development and debugging phases of a project), or it can be run on a remote machine to benefit from the distributed processing power and multi-user access features of a shared network server. The component does not need to be recompiled or changed in any way to support this location independence.

The server in this project uses a visible form purely for demonstration purposes. There is no functional need for the server to have any UI, though state information presented through a UI can help with debugging or run-time monitoring needs. (State monitoring requirements can also be addressed by providing a status method on the server that can be queried by a monitor application.)

File	Description
Book_cli.vbp	Client component project file.
Book_cli.frm	Client main form.
Book_cli.frx	Client main dialog graphics file.
Frmchart.frm	Client result chart form.
Frmcogs.frm	Client cost of goods form.
Book_cli.bas	Client main/global utility module.
Book_svr.vbp	Server component project file.
Book_svr.frm	Server status indicator form.
Book_svr.frx	Server status indicator graphics file.
Book_svr.bas	Server main/global utility module.
Sales.cls	Server main/public class module.
Model.cls	Server private business model rule class module.
Taxes.cls	Server private tax rule class module.
Booksale.mdb	Sample Jet database file.
Booksale.ldb	Sample database file created at run time by Jet.
Booksale.txt	Text file containing overview information on this sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Book\_svr.vbp file, which is listed in the \Samples\Clisvr\Booksale subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the server project.

Once running, this project temporarily registers its classes in the system registry. You can then start a second instance of Visual Basic to run Book\_cli.vbp, the user interface component that uses the class modules defined in Book\_svr.vbp.

**Note** The component can be run on the same machine as the user interface component, or it can be run on a remote machine to benefit from distributed processing power and support multi-user access. The component server does not need to be recompiled or changed in any way to support this location independence.

## CallDlls (CallDlls.vbp)

This sample demonstrates calling procedures in Dynamic-link libraries (DLLs). For more information, see Chapter 26, "Calling Procedures in DLLs," of the *Programmer's Guide*.

<b>File</b>	<b>Description</b>
Calldlls.frm	The main form for the application.
Calldlls.frx	Binary data for the Calldlls.frm file.
Calldlls.vbp	The project file for the application.
Declares.bas	Module containing declarations of DLLs used in the sample.
Frmmenus.frm	Form for showing system information.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Calldlls.vbp file, which is listed in the \Samples\CompTool\CallDlls subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.



## Coffee (Coffee2.vbp, CoffWat2.vbp, MTCoffee.vbp, XTimers.vbp)

Coffee consists of a client, CoffWat2, and two ActiveX components (OLE servers), Coffee2 and MTCoffee. Together, these three projects demonstrate:

- Asynchronous notifications using events (Coffee2).
- Asynchronous notifications using call-back methods (Coffee2).
- Multithreading (MTCoffee).

XTimers.vbp is a helper project that provides a code-only timer used by Coffee2 and MTCoffee.

This is an expanded version of the project developed in the step-by-step procedures in Chapter 3, "Creating an ActiveX Exe Component," of *Creating ActiveX Components*. For more information on asynchronous notifications and multithreading, see Chapter 8, "Building Code Components."

File	Description
<b>CoffWat2.vbp</b>	The CoffeeWatch client project.
CWMod1.bas	Startup code for the project.
CWForm1.frm	The client's main form.
ICoffNot.cls	The ICoffeeNotify interface used for call-back methods..
CWNotMe.cls	NotifyMe object implements the ICoffeeNotify interface, so it can receive call-backs.
CWThread.frm	Client form for demonstrating multithreading.
CWCoTrk.cls	The CoffeeTracker object is used to wait for completion events from the multithreaded Coffee object.
AboutCof.txt	A copy of this document, demonstrating the ability to store related documents in a project.
<b>Coffee2.vbp</b>	The Coffee2 component project.
Co2Cmon.cls	The CoffeeMonitor class used to demonstrate asynchronous notifications using events.
Co2Conn.cls	The Connector class that enables multiple clients to share a CoffeeMonitor object.
Co2CMon2.cls	The CoffeeMonitor2 class used to demonstrate asynchronous notifications using call-back methods.
Co2Conn2.cls	The Connector2 class that enables multiple clients to share a CoffeeMonitor2 object.
Co2Mod1.bas	Holds a reference to the shared CoffeeMonitor and CoffeeMonitor2 objects.
<b>MTCoffee.vbp</b>	The MTCoffee component project.
MTCoffee.cls	The multithreaded <b>Coffee</b> object.
MTCMod1.bas	A standard module for demonstrating instancing of global data.
<b>XTimers.vbp</b>	The call-back timer project.
XTimer.cls	The XTimer object.
XTimerS.bas	Support module for XTimer.

### To Run

Although there's not a lot of code in it, this is a complex sample to run, because it demonstrates out-of-process components. For debugging, an out-of-process component must be run in a separate copy of Visual Basic.

In addition, the sample demonstrates features — multithreading and code-only timers — that either cannot be demonstrated in, or are dangerous to run in, the development environment.

- 1 Load XTimers.vbp into Visual Basic. On the **File** menu, select **Make XTimers.dll** to make the project into a .dll file.  
DO NOT run XTimers in the development environment at this time. XTimers must be compiled because it uses Windows APIs for a code-only timer object.
- 2 Load MTCoffee into Visual Basic. On the **Project** menu, select **References** to open the References dialog box. Select **XTimers** and click **OK**, to set a reference to the compiled DLL.
- 3 On the **File** menu, select **Make MTCoffee.exe** to make the project into an .exe file.  
DO NOT run MTCoffee in the development environment at this time. You need to make the project into an executable because the development environment can't support multiple threads of execution — if you run MTCoffee within Visual Basic, it won't display multithreading behavior.
- 4 Load Coffee2.vbp into Visual Basic. On the **Project** menu, select **References** to open the References dialog box. Select **XTimers** and click **OK**, to set a reference to the compiled DLL.
- 5 Press CTRL+F5 to run the Coffee2 project.  
Use CTRL+F5 instead of F5 when running an out-of-process component project, to ensure that all compile errors are caught before the component is supplying objects to the client. (See "Creating an ActiveX Exe Component," in Books Online.)
- 6 Start another instance of Visual Basic, and load CoffWat2.vbp. On the **Project** menu, select **References** to open the References dialog box. Select **Coffee2** (make sure you get the entry for Coffee2.vbp) and **MTCoffee** (make sure you get the entry for MTCoffee.exe), and then click **OK**, to set references to the components.
- 7 Press F5 to run the project.

After running the sample application and observing the behavior of MTCoffee when it allocates a separate thread for each Coffee object, you may find it instructive to recompile MTCoffee after changing the threading to a round-robin thread pool. (You can find this option in the Unattended Execution box, on the General tab of the Project Properties dialog box.) When using a round-robin thread pool, you can observe blocking and global data sharing between objects on the same thread.

You can also compile Coffee2.exe and CoffWat2.exe, and run multiple clients, to observe sharing of the asynchronous notification objects CoffeeMonitor and CoffeeMonitor2. With multiple clients, the round-robin thread pool means blocking and data sharing between objects in use by different clients.

To understand what's going on in this sample, see "Building Code Components," in *Creating ActiveX Components* in Books Online.

## Control Plus (CtlPlus.vbp)

CtlPlus demonstrates a number of features of Visual Basic-Authored ActiveX Controls.

This is an expanded version of the project developed in the step-by-step procedures in Chapter 4, "Creating an ActiveX Control," in *Creating ActiveX Components*. For more information control creation, see Chapter 9, "Building ActiveX Controls," in *Creating ActiveX Components*.

File	Description
<b>CtlPlus.vbg</b>	The program group.
<b>CtlPlus.vbp</b>	The control component project.
CPSShapeL.ctl	The ShapeLabel control.
CPSLGen.pag	The General property page tab for ShapeLabel.
<b>TestCtlP</b>	The test program.
TCPForm1.frm	The test form for displaying ShapeLabel.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the program group, CtlPlus.vbg file, which is listed in the \Samples\CompTool\ActvComp\CtlPlus subdirectory of the main Visual Basic directory.

Open Form1 in TestCtlP, to view the design-time behavior of ShapeLabel.

Press CTRL+F5 or choose **Start With Full Compile** from the **Run** menu to run the program group, and view the control's run-time behavior.

## Controls (Controls.vbp)

This sample application demonstrates the use of Visual Basic controls such as the **TextBox**, **CommandButton**, **Image** and others. The sample illustrates the usage of many standard properties.

### Background

The sample is comprised of several forms, each with a control or set of controls on it. Controls demonstrated include the following: **TextBox**, **HScrollbar**, **ListBox**, **PictureBox**, **OptionButton**, **CommandButton** and **Label** control. For more information, see Chapter 3, "Creating and Using Controls," in the *Programmer's Guide*.

File	Description
Button.frm	Form demonstrating the usage of CommandButton controls.
Button.frx	Binary data file for the Button.frm file.
Check.frm	Form demonstrating usage of a CheckButton control.
Controls.vbp	The project file for the sample application.
Images.frm	Form demonstrating usage of the Image and Shape controls.
Images.frx	Binary data file for the Image.frm file.
Main.frm	The main form for the sample application.
Multi.frm	Form demonstrating usage of the ListBox control.
Options.frm	Form demonstrating usage of the OptionButton and Frame controls.
Text.frm	Form demonstrating usage of the TextBox control.
Text.frx	Binary data file for the Text.frm file.
Wordwrap.frm	Form demonstrating usage of the Label control.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Controls.vbp file, which is listed in the \Samples\PGuide\Controls subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## DataTree (Datatree.vbp)

The DataTree sample application demonstrates how to use the **TreeView** control and the **ListView** control to view the contents of a small database (Biblio.mdb). The **ProgressBar** control is also used to give the user a visual indication of how long a process is taking.

<b>File</b>	<b>Description</b>
TreeView.frx	Binary data for the TreeView.frm
TreeView.frx	Form containing the <b>TreeView</b> , <b>ListView</b> , and <b>Progressbar</b> controls.
Datatree.vbp	Main project for the sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the DataTree.vbp file, which is listed in the \Samples\CompTool\DataTree subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## Errors (Errors.vbp)

This sample application demonstrates various Visual Basic error handling techniques, including examples of inline and centralized error handling.

### Background

The sample is comprised of a single form with a set of command buttons to invoke error handling procedures, plus a basic module containing a central error handling routine. By stepping through the code in the procedures you can see the effects of different error handling methods. For more information, see Chapter 13, "Debugging Your Code and Handling Errors," in the *Programmer's Guide*.

File	Description
Errors.bas	Module containing centralized error handling code.
Errors.frm	Form demonstrating error handling techniques.
Errors.vbp	The project file for the sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Errors.vbp file, which is listed in the \Samples\Pguide\Errors subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## FileCtls (FileCtls.vbp)

This sample application allows the user to search a directory for specific files, or files with a particular extension. The sample uses file-system controls such as the **File ListBox**, **Drive ListBox**, and **Directory ListBox** controls.

<b>File</b>	<b>Description</b>
Seek.frm	The main form for the sample.
Winseek.vbp	The project file for the sample.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Winseek.vbp file, which is listed in the \Samples\Misc\FileCts subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## FirstApp (FirstApp.vbp)

This sample application demonstrates the usage of the Data Control with other data aware controls. It provides a simple introduction to assigning properties at run-time.

### Background

The sample is comprised of a single form with a **DBGrid** control connected to a **Data** control. For more information, see Chapter 2, "Developing an Application in Visual Basic," in the *Programmer's Guide*.

File	Description
Form1.frx	Binary data for Form1.frm
Form1.frm	
Firstapp.vbp	The project file for the application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Firstapp.vbp file, which is listed in the \Samples\Pguide\Firstapp subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.



## GeoFacts (GeoFacts.vbp)

The GeoFacts sample application demonstrates how to work with objects provided by other applications. In this case, the objects are provided by Microsoft Excel using the **GetObject** function. Using the reference returned by the function, two **ComboBox** controls and a **ListBox** control are dynamically filled with data from an Excel spreadsheet.

<b>File</b>	<b>Description</b>
World.xls	Excel spreadsheet that supplies data for the application.
Geofacts.frm	Form for the application.
Geofacts.bas	Code module containing procedures for the application.
Geofacts.vbp	The project file for the application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the GeoFacts.vbp file, which is listed in the \Samples\PGuide\GeoFacts subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## ListCmbo (ListCmbo.vbp)

This sample program demonstrates data binding to the standard **ListBox** and **ComboBox** controls.

### Background

Visual Basic 5.0 allows you to bind the standard **ListBox** and **ComboBox** controls to a **Data** control. Unlike other bound controls, the **ListBox** and **ComboBox** controls require that you populate the List portions of the control before binding in order to correctly display the current record's value in the control.

When binding to a **ComboBox** and **ListBox** control, the **Data** control selects the current record's value from the list of available entries within the control. For example, binding the **ListBox** control to a field within a database which contains state abbreviations, the **Data** control will search the values within the **ListBox** and select the matching entry. As a result, you must populate the control with all possible state abbreviations before binding the control. If the **ListBox** is not populated, the **Data** control will not be able to find the matching entry and no item will be selected.

This sample application demonstrates this behavior and shows how to populate the **ListBox** and **ComboBox** controls prior to binding. A single form is used to display the Publishers table contained within Biblio.mdb. Publishers has a "state" field which contains the state abbreviation of the current Publisher's address. At runtime, the sample performs a query of the Publishers table of all distinct values contained within the state field and populates a **ListBox** and **ComboBox** control. Then, the controls are bound to the **Data** control.

When the user cycles through the Publishers records using the navigation buttons on the **Data** control, the "state" field value for the current record is selected in the **ListBox** and **ComboBox** controls. The user can then choose which of these controls to view by clicking on the appropriate option buttons.

File	Description
Frmpub.frm	The main Publishers form, which displays the Publishers records.
Frmpub.frx	The binary portion of the Publishers form.
Frmabout.frm	Standard About form.
Frmabout.frx	The binary portion of the About form.
Listcmbo.vbp	The project file for the sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the ListCmbo.vbp file, which is listed in the \Samples\PGuide\Liscmbo subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

To cycle through the records, click on the navigation buttons on the **Data** control. To view either the **ListBox** or **ComboBox** controls, click on the appropriate option buttons.

## MDI (MDINote.vbp)

This sample application demonstrates a Multiple Document Interface (MDI) notepad application. It also provides an introduction to menu creation.

### Background

The sample is comprised of a parent form, a child form, a modal dialog form, and two basic modules. You can use this sample application in conjunction with the SDINote sample application to understand the differences between interface styles. For more information, see Chapter 6, "Creating a User Interface," in the *Programmer's Guide*.

File	Description
Filopen.bas	Module containing common file handling code.
Find.frm	Modal dialog form.
Mdi.frm	The MDI parent form.
Mdi.frx	Binary data file for the Mdi.frm file.
Mdinote.bas	Module containing shared code.
MDINote.vbp	The MDI note project file.
Notepad.frm	The MDI child form.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Mdinote.vbp file, which is listed in the \Samples\Pguide\MDI subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## MSFlexGd (MSFlexGd.vbp)

The MSFlexGd sample application demonstrates the **MSFlexGrid** control.

<b>File</b>	<b>Description</b>
Flex.frx	Binary information for the Flex.frm file
Flex.frm	The main form which displays the control.
Flex.vbp	The project file for the application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the MSFlexGd.vbp file, which is listed in the \Samples\CompTool\MSFlexGd subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

To cycle through the records, click on the navigation buttons on the **Data** control. To view either the **ListBox** or **ComboBox** controls, click on the appropriate option buttons.

## OleCont (OleCont.vbp)

This sample application demonstrates the capabilities of the OLE **Container** control. The sample allows you to save or open OLE objects using the **SaveToFile** and **ReadFromFile** methods. For more information, see Chapter 8, "Using the OLE Container Control," in the *Programmer's Guide*.

<b>File</b>	<b>Description</b>
Contchld.frm.	An MDI child form containing the OLE <b>Container</b> control
Contmdi.frm	The main form.
Olecont.bas	Module containing Public Sub and Function procedures.
Colecont.vbp	The project file for this sample.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Olecont.vbp file, which is listed in the \Samples\PGuide\Olecont subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

# Optimize (Optimize.vbp)

This sample application demonstrates several optimization techniques for Visual Basic. Speed comparisons are made between various coding techniques to illustrate performance differences.

## Background

The sample is comprised of several forms, each demonstrating optimized and non-optimized code. When running this sample, it's a good idea to shut down any other running applications in order to get accurate timings. For more information, see Chapter 15, "Designing for Performance and Compatibility," in the *Programmer's Guide*.

File	Description
About.frm	About box form for application.
About.frx	Binary data file for the ABOUT.FRM file.
Algorithm.frm	Form demonstrating algorithms for drawing.
Code.frm	Form demonstrating code optimizations.
Collection.frm	Form demonstrating optimizations for collections.
Explore.frm	The main form for the sample application with an Explorer style interface.
Explore.frx	Binary data file for the EXPLORE.FRM file.
Frmimage.frm	Auxiliary form for the Splash Screen demonstration.
Frmimage.frx	Binary data file for the FRMIMAGE.FRM file.
FrmPics.frm	Form demonstrating optimizations for displaying images.
FrmPics.frx	Binary data file for the FRMPICS.FRM file.
FrmShow1.frm	Form demonstrating optimizations for forms.
FrmShow2.frm	Auxiliary form demonstrating optimizations for forms.
Global.bas	Module containing shared code.
Graphics.frm	Form demonstrating optimizations for graphics methods.
Graphics.frx	Binary data file for the GRAPHICS.FRM file.
Numbers.frm	Form demonstrating data type optimizations.
Optimize.vbp	Project file for this application.
Paintpic.frm	Form demonstrating painting optimizations.
Paintpic.frx	Binary data file for the PAINTPIC.FRM file.
Splash.frm	Sample splash screen.
Splash.frx	Binary data file for the SPLASH.FRM file.
Splshdmo.frm	Form demonstrating startup optimizations.
Strings.frm	Form demonstrating string handling optimizations

## To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Optimize.vbp file, which is listed in the \Samples\Pguide\Optimize subdirectory of the main Visual Basic directory. Press F5 or

choose **Start** from the **Run** menu to run the application.

## PalMode (Palettes.vbp)

This sample application demonstrates the effects of different PaletteMode settings on 256 color images. It also provides an introduction to the **Picture** object.

### Background

This sample is comprised of a single form for displaying images. In order to see the effects of the **PaletteMode** property, it is necessary to run this sample on a display set to 256 color mode. For more information, see Chapter 12, "Working with Text and Graphics," in the *Programmer's Guide*.

File	Description
Banner.gif	GIF image file.
Clouds.bmp	Bitmap image file.
Forest.jpg	JPG image file.
Palettes.frm	Main form for application.
Palettes.frx	Binary data file for the Palettes.frm file.
Palettes.vbp	The Palettes project file.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Palettes.vbp file, which is listed in the \Samples\Pguide\Palmode subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.



## PicClip (RedTop.vbp)

This sample program demonstrates one of the possible uses of the **PictureClip** control. This sample application uses the **PictureClip** control to spin a top.

<b>File</b>	<b>Description</b>
Infoform.frm	The information form.
Infoform.frx	The binary portion of the information form.
Redtop.frm	The main form.
Redtop.frx	The binary portion of the main form.
Redtop.vbp	The project file for the sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Redtop.vbp file, which is listed in the \Samples\CompTool\PicClip subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## Programming with Objects (ProgWOb.vbp)

ProgWOb.vbp demonstrates concepts from Chapter 9, “Programming with Objects,” in the *Visual Basic Programmer’s Guide*, including creating collection classes, events, the Implements statement, polymorphism, and Friend procedures.

File	Description
<b>ProgWOb.vbp</b>	The project.
PWOMain.frm	The main form, from which you can select demos.
PWOFrien.frm	First demo: Using Friend procedures to pass user-defined types between objects.
PWOTestC.cls	Defines the TestClass object used to demonstrate Friend procedures.
PWOFrien.bas	Defines the user-defined type passed between TestClass objects.
PWOCYOCC.frm	Second demo: From this form you can run the three code examples from Creating Your Own Collection Classes.
PWOEmpl.cls	Employee class used by all three examples.
PWOSTraw.frm	House of Straw example: Using a public Collection object.
PWOSBus1.cls	SmallBusiness1 class used in House of Straw example.
PWOSTick.frm	House of Sticks example: Using a private Collection object.
PWOSBus2.cls	SmallBusiness2 class used in House of Sticks example.
PWOBrick.frm	House of Bricks example: Creating your own collection class.
PWOEmpls.cls	Employees collection class used in House of Bricks example. Can be used with For Each.
PWOSBus3.cls	SmallBusiness3 class used in House of Bricks example.
PWOEvent.frm	Third demo: Shows two things: (1) an object performing a long task can raise events to notify the caller of its progress, and (2) events can be broadcast to multiple receivers.
PWOWidg	Widget object uses events to notify caller of progress during a long task.
PWOEvRec.frm	Receiver form handles an event, and shows what happens when ByRef arguments of an event are modified by receivers.
PWOImple	Fourth demo: Shows how polymorphism allows early binding with multiple classes of objects, and illustrates the performance impact.
PWOIShap	Defines the IShape interface implemented by all other classes in the demo.
PWOPoly	Polygon is a very crude object for drawing

	polygons. Implements IShape for drawing.
PWORect	Rectangle object implements IShape for drawing. It also implements the Polygon interface, and uses an inner Polygon object to store its data.
PWOTrngl	Triangle object implements IShape and Polygon. It delegates to an inner Polygon object for both IShape and most Polygon functions.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the ProgWOb.vbp file, which is listed in the \Samples\Pguide\ProgWOb subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## Sdi (SdiNote.vbp)

This sample application demonstrates a Single Document Interface (SDI) notepad application. It also provides an introduction to menu creation and toolbars.

### Background

The sample is comprised of a main form, a modal dialog form, and two basic modules. You can use this sample application in conjunction with the MDI Note sample application to understand the differences between interface styles. For more information, see Chapter 6, "Creating a User Interface," in the *Programmer's Guide*.

File	Description
Filopen.bas	Module containing common file handling code.
Find.frm	Modal dialog form.
FrmSDI.frm	The main notepad form.
Frmsdi.frx	Binary data file for the Frmsdi.frm file.
SDI Note.bas	Module containing shared code.
SDI Note.vbp	The SDI Note project file.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Sdinote.vbp file, which is listed in the \Samples\Pguide\SDI subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## TabOrder (TabOrder.vbp)

The TabOrder sample application uses the VB5 extensibility model, as well as traditional VB techniques, to reset the tab order of a given form. Many times when creating forms in VB, the controls end up in an odd tab order. The TabOrder application allows you to reset the tab order to top to bottom, or left to right order. You can also move items in the TabOrder's list up or down, then apply the changes to the form. See Chapter 3, "How to Build an Add-In," in "Extending the Visual Basic Environment Using Add-Ins," of the *Component Tools Guide*.

File	Description
Tab.bmp	Bitmap for the TabOrder application.
Tab.ico	Icon for the TabOrder application.
TabOrder.rc	Resource file for the TabOrder application.
TaOrder.cls	Class file for the TabOrder application.
TabOrder.frx	Binary data for the TabOrder.frm file.
Taborder.frm	The main form for the application.
Taborder.bas	Code module for the application.
Taborder.vbp	The project file for the application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Taborder.vbp file, which is listed in the \Samples\CompTool\AddIns\Taborder subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## UnBndGrd (UnBndGrd.vbp)

The UnBound DBGrid sample application is a simple program that illustrates how to use the **DBGrid** control in unbound mode (**DataMode** = 1).

### Background

In most cases, the **DBGrid** control can be bound to the **Data** control. Some situations, however, may require you to use **DBGrid** in unbound mode, such as with proprietary database formats or simple data sets which aren't supported by the **Data** control. This sample makes use of the UnBound events to display sample data which is stored in an array.

File	Description
UnBndGrd.frm	The main form with the <b>DBGrid</b> control on it.
UnBndGrd.frx	The binary portion of the UnBndGrd.frm file.
UnBndGrd.vbp	The project file for the sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the UnBndGrd.VBP file, which is listed in the \Samples\Misc\UnBndGrd subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## VBMail (VBMail.vbp)

This sample program demonstrates the use of the MAPI controls by sending and receiving electronic mail. It uses the messaging application programming interface (MAPI) controls: MAPI messages and MAPI session.

**Note** To run this sample, you need a MAPI-compliant messaging system. If you are using Windows 95, you will need Microsoft Exchange.

### Background

The MAPI session control establishes a MAPI session and signs off from a MAPI session. The MAPI messages control allows you to perform a variety of messaging systems functions after a messaging session has been established. These functions include accessing, downloading, and sending messages, displaying the details and address book dialog boxes, accessing data attachments, resolving recipient names during addressing, and performing compose, reply, reply-all, forward, and deleting actions on messages.

File	Description
Mailst.frm	Form for displaying the list of mail messages.
Mailst.frx	Contains binary data for the sample.
Mailoptf.frm	Form for setting options.
Mailsup.bas	Module of support procedures for the example.
MsgView.frm	Form in which a viewed message is displayed.
Newmsg.frm	Form for creating a new message.
Vbmail.frm	The main form.
Vbmail.vbp	The project file for this sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the VBMAIL.VBP file, which is listed in the \Samples\CompTool\VBTerm subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## VBTerm (VBTerm.vbp)

VBTerm is a terminal emulation program using the **MSComm** control. The sample demonstrates how to use the **MSComm** control with a serial port.

### Background

The **MSComm** control allows you to open a serial port, change its settings, send and receive data through the port, and monitor and set many of the different data lines. Its dual-method access allows for both polling and event-driven communications.

File	Description
Cansend.frm	Dialog box used during file transfer.
Termset.frm	Form used to change the serial port settings.
Termset.frx	Binary portion of TERMSET.FRM.
Vbterm.frm	The main form.
Vbterm.frx	Binary portion of VBTERM.FRM.
Vbterm.glo	Public declarations.
Vbterm.vbp	The project file for the sample application.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the VBTerm.vbp file, which is listed in the \Samples\CompTool\VBterm subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.



## VCR (Vcr.vbp)

This sample application demonstrates how Visual Basic classes can be modeled after real-world objects such as a VCR.

### Background

This sample is comprised of a main form used to represent a VCR, and two supporting classes representing the tape transport and recorder objects. For more information, see Chapter 5, "Programming Fundamentals," in the *Programmer's Guide*.

File	Description
Recorder.cls	Class module for recorder object.
Set.frm	Modal form for setting recorder properties.
Tape.cls	Class module for tape transport object.
Vcr.bas	Module with shared code.
Vcr.frm	Main form for application.
Vcr.frx	Binary data file for the VCR.frm file.
Vcr.vbp	The VCR project file.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Vcr.vbp file, which is listed in the \Samples\Pguide\Vcr subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the application.

## VisData (Visdata.vbp)

This sample program demonstrates various programming techniques used to access data through the Data Access Object (DAO) layer built into the Visual Basic Professional Edition.

**Note** To access ODBC data sources with this sample, you must first install ODBC using the ODBC setup program provided with Visual Basic, Professional Edition.

VisData behaves like a general purpose database utility, capable of the following functions:

Database and table creation.

- Table modification (adding and deleting fields and indexes).
- Data browsing/modifying using the three recordset types (table, dynaset, and snapshot) and three form types (single record, data control, and data grid).
- Property browsing on all objects.
- Form demonstrating the new data-bound list and data-bound combo box controls.
- Import/export to all supported data types.
- Direct SQL statement execution for any SQL supported functions such as **Insert, Update, Delete, Drop, Create, Dump**, and so forth.
- AdHoc Query builder that helps users unfamiliar with SQL to create complex queries with Where clauses, Joins, Order By, and Group By expressions while limiting output to selected columns.
- Transaction processing.
- Copying table structures and data to the same or different database.
- Support of Jet MDB, Dbase III and IV, FoxPro 2.x, Paradox 3.x and 4.x, Btrieve, Text, Excel, and SQL Server, both DDL and DML.
- QueryDef creation, modification, and execution.
- JET security creation/modification.
- Relation/referential Integrity creation and modification.
- Management of attached tables.
- Use as an add-in to Visual Basic (see VISDATA.TXT).

The code contains comments to help explain the use of the various methods in the data access layer. Code and forms may be copied from this application to other applications with minimal modification.

File	Description
Aboutbox.frm	Standard About box for the application.
Addfield.frm	Form to add fields to tables.
Addindex.frm	Form used to add indexes to Tables.
Attach.frm	Attached table list.
Cpystru.frm	Form to copy table structures.
Database.frm	Form to select a database.
Dataform.frm	Data control form.
Datagrid.frm	Form used to display data in a data-bound grid control.
Datatype.frm	Data type selector for import/export utility.
Dbpwd.frm	Form to enter a password.
Dcprop.frm	Data control property form.
Dfd.frm	Data Form Designer add-in form.
Dynasnap.frm	Form to display data in single-record mode.
Errors.frm	Errors collection form.

Expname.frm	Export name prompt.
Find.frm	Form used to find records in a Dynaset.
Grpsusrs.frm	Jet security form.
Imptbls.frm	Import tables list.
Join.frm	Form used to add joins to the Query Builder.
Loginfrm.frm	Jet WorkSpace logon dialog box.
Newattch.frm	New attachment form.
Newpw.frm	New password dialog.
Newug.frm	New User/Group dialog.
Odbclogn.frm	ODBC logon form.
Property.frm	Property browser.
Query.frm	Form used to build queries.
Replace.frm	Form to perform global replace operations on a table.
Seek.frm	Form used to get input for the <b>Seek</b> function.
Sql.frm	Form to enter and execute <b>SQL</b> statements.
Tableobj.frm	Form used to display data in a <b>Table</b> object.
Tblstru.frm	Form to display and modify table structures.
Vbimex.frm	Import/export form.
Vdclass.cls	Class module for VisData.
Vdmdi.frm	Main MDI form for the application.
Visdata.bas	Support functions for the application.
Visdata.vbp	Project file for the application.
Zoom.frm	Form to zoom in on character data.
Vb5.hlp	VisData Help File.
*.frx files	Contains binary data for the associated .frm File.
*.ico files	Supplied for modification by user.

## To Run

On the **File** menu, click **Open Project**, and select the Visdata.vbp file, which is located in the \Samples\Visdata subdirectory of the main Visual Basic directory.

If you want to open a local database, you simply need to choose the type of database, and a **File Open** dialog box will be displayed with the file type set to the requested data file type.

If you choose ODBC on the **Open Database** submenu, you will then see the Open DataBase form. Since you probably have no servers entered, you will need to enter a name for an existing SQL server on your network. If you already know the user ID and password, you can add them as well. The Database name is optional. Once you have entered this data, choose **OK**, and you should be able to log on to the server. You may get some more dialogs in the process. Answer any questions you can and ask the SQL administrator for help if you run into problems or don't know some of the parameters.

Once a database is open, double-clicking a table name will open the table in the selected form type and recordset type. Use the Query Builder to create dynasets with selected data from one or more tables at a time.

The table-type Recordset is always updatable. The dynaset-type Recordset will be updatable in most cases except on ODBC with no unique index, certain multiple table joins, and other SQL Select statements, such as count(\*), max(), and so forth. The snapshot-type Recordset is never updatable.

## Callback Sample Application (Clbk\_Svr.vbp and Clbk\_Cli.vbp), Enterprise Edition

This sample demonstrates the use of an ActiveX object pointer being passed to an external (and optionally remote) ActiveX server component. The server then periodically calls a method on this object pointer. This has the effect of a server-initiated callback to the client, which can be a much better application model than the polling a client application might have to do otherwise to find the status of a server. Although this demo simply returns the time to the client, it could just as easily return data, news, or other information it has been told the client wants to know. The benefit here is that the server does all the work looking for data the client might need and the client does other work, only being interrupted when the server actually has something that interests it.

The server in this project uses a visible form purely for demonstration purposes. There is no functional need for the server to have any UI, though state information presented through a UI can help with debugging or run-time monitoring needs. (Note, state monitoring requirements could also be addressed by providing a status method on the server that could be queried by a monitor application.)

File	Description
Clbk_cli.vbp	Client component project file.
Clbk_cli.frm	Client main form.
Clbk_cli.bas	Client main/global utility module.
Clbk_cli.cls	Client class module.
Clbk_svr.vbp	Server component project file.
Clbk_svr.frm	Server status indicator form.
Clbk_svr.bas	Server main/global utility module.
Clbk_svr.cls	Server class module.
Callback.txt	A text file containing a project overview and description.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Clbk\_svr.vbp file, which is listed in the \Samples\CliSrv\Callback subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the server project.

Once running, this project temporarily registers its classes in the system registry. You can then start a second instance of Visual Basic to run Clbk\_cli.vbp, the client that uses the class modules defined in Clbk\_svr.vbp.

**Note** The Automation server can be run on the same machine as the client, or it can be run on a remote machine to benefit from distributed processing power and support multi-user access. The Automation server does not need to be recompiled or changed in any way to support this location independence.

## Hello World Remote Automation Sample Application (Helo\_Svr.vbp and Helo\_Cli.vbp), Enterprise Edition

This 6-line Hello World sample demonstrates one of the simplest possible examples of Remote Automation and DCOM. The application runs on both a single machine and in a 2-machine distributed configuration without needing to be recompiled.

The server in this project uses a visible form purely for demonstration purposes. There is no functional need for the server to have any UI, though state information presented through a UI can help with debugging or run-time monitoring needs. (Note, state monitoring requirements could also be addressed by providing a status method on the server that could be queried by a monitor application.)

File	Description
Helo_cli.vbp	Client component project file.
Helo_cli.frm	Client main form.
Helo_svr.vbp	Server component project file.
Helo_svr.frm	Server status indicator form.
Helo_svr.cls	Server class module.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Helo\_svr.vbp file, which is listed in the \Samples\CliSrv\Hello subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the server project.

Once running, this project temporarily registers its classes in the system registry. You can then start a second instance of Visual Basic to run Helo\_cli.vbp, the client that uses the class modules defined in Helo\_svr.vbp.

**Note** The Automation server can be run on the same machine as the client, or it can be run on a remote machine to benefit from distributed processing power and support multi-user access. The Automation server does not need to be recompiled or changed in any way to support this location independence.

## Interface Sample Application (Intr\_Svr.vbp and Intr\_Cli.vbp), Enterprise Edition

This sample demonstrates a way to make more efficient user of COM's "apartment model" resource allocation algorithm when the context of an entire project needs to be preserved for a specific client.

The server in this project does not have a form module, and as such, will not be visible on the machine it is running on. (When a client has an active reference to it, however, it can be seen in the running task list.)

File	Description
Intr_cli.vbp	Client component project file.
Intr_cli.frm	Client main form.
Intr_svr.bas	Server main/global utility module.
Intr_svr.cls	A public interface class module that clients use to access the private classes of this component.
Getdate.cls	A private class that returns the current date.
Gettime.cls	A private class that returns the current time.
Intrface.txt	A text file with a project overview and description.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Intr\_svr.vbp file, which is listed in the \Samples\CliSrv\Intrface subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the server project.

Once running, this project temporarily registers its classes in the system registry. You can then start a second instance of Visual Basic to run Intr\_cli.vbp, the client that uses the class modules defined in Intr\_svr.vbp.

**Note** The Automation server can be run on the same machine as the client, or it can be run on a remote machine to benefit from distributed processing power and support multi-user access. The Automation server does not need to be recompiled or changed in any way to support this location independence.

## Passthrough Server Sample Application (Pass\_Svr.vbp and Pass\_Cli.vbp), Enterprise Edition

This sample application is an example of a simple pass-through server. It has one method called **RunServer**, which takes a ProgID parameter of the server that is to be started. The server specified by the ProgID is run on the same machine that the pass-through server is running on. If the server specified by ProgID is an in-process server, it is run in the same process space as the pass-through server.

**RunServer** returns the handle of the started server so that the client can talk directly to the requested server. Because the pass-through server is a single-use server, every client that uses it will get its own instance of the pass-through server, along with its own process space and thread of execution.

By using the pass-through server, client applications can create their own independent execution context on a remote server and then start and stop in-process servers within that context as needed. Because in-process servers can be started very quickly and because they are run within the pass-through server's process space, this is a much more efficient execution model than solely using out-of-process servers.

The server in this project uses a visible form purely for demonstration purposes. There is no functional need for the server to have any UI, though state information presented through a UI can help with debugging or run-time monitoring needs. (State monitoring requirements can also be addressed by providing a status method on the server that can be queried by a monitor application.)

File	Description
Pass_cli.vbp	Client component project file.
Pass_cli.frm	Client main form.
Pass_svr.vbp	Server component project file.
Pass_svr.frm	Server status indicator form.
Pass_svr.bas	Server main/global utility module.
Pass_svr.cls	Server class module.
Passthru.txt	A text file containing a project overview and description.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Pass\_svr.vbp file, which is listed in the \Samples\CliSrv\Passthru subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the server project.

Once running, this project temporarily registers its classes in the system registry. You can then start a second instance of Visual Basic to run Pass\_cli.vbp, the client that uses the class modules defined in Pass\_svr.vbp.

**Note** The Automation server can be run on the same machine as the client, or it can be run on a remote machine to benefit from distributed processing power and support multi-user access. The Automation server does not need to be recompiled or changed in any way to support this location independence.

## Pool Manager Sample Application (Pmgr\_Svr.vp and Pmgr\_Cli.vbp), Enterprise Edition

This sample app is an example of a simple pool manager that could be used to maintain open instances of Automation servers for client applications. Using this scheme, client applications ask the pool manager for a pointer to an object they want to use. The pool manager checks the pool, and either grants or denies the request. This approach has a number of benefits:

- It avoids the lengthy Automation server creation costs associated with each client request. This is because the pool is typically created before a client needs a server.
- Based on the frequency of client requests, the potential number of clients, and the duration of server tasks, a pool size can be created (or even adjusted as demand dictates at various times) that is significantly smaller than a one-to-one allocation scheme for clients and servers. For example: if the number of clients is 60, the average frequency of requests is one per minute per client, and the average duration of a server task is one second. A linear allocation plan would suggest that one server is needed to meet the needs of 60 clients. Realistically, requests will not be perfectly spaced; therefore, a reasonable level of redundancy, for instance a pool size of 5 Automation servers, could be used to meet the normal needs of the 60 clients.
- It limits the number of servers that can be created of a specific type to a threshold that has been determined by the server administrator. This can be a very useful tuning parameter, and can also be useful in preventing the server from being abused by a peak request of a low priority server.

The server in this project uses a visible form purely for demonstration purposes. There is no functional need for the server to have any UI, though state information presented through a UI can help with debugging or run-time monitoring needs. (State monitoring requirements can also be addressed by providing a status method on the server that can be queried by a monitor application.)

File	Description
Pmgr_cli.vbp	Client component project file.
Pmgr_cli.frm	Client main form.
Pmgr_svr.vbp	Server component project file.
Pmgr_svr.frm	Server status indicator form.
Pmgr_svr.bas	Server main/global utility module.
Pmgr_svr.cls	Server class module.
Poolmgr.txt	A text file containing a project overview and description.

### To Run

From the Visual Basic **File** menu, choose **Open Project** and select the Pmgr\_svr.vbp file, which is listed in the \Samples\CliSrv\Poolmgr subdirectory of the main Visual Basic directory. Press F5 or choose **Start** from the **Run** menu to run the server project.

Once running, this project temporarily registers its classes in the system registry. You can then start a second instance of Visual Basic to run Pmgr\_cli.vbp, the client that uses the class modules defined in Pmgr\_svr.vbp.

**Note** The Automation server can be run on the same machine as the client, or it can be run on a remote machine to benefit from distributed processing power and support multi-user access. The Automation server does not need to be recompiled or changed in any way to support this location independence.



# Application Performance Explorer (APE) Sample Application Suite, Enterprise Edition

This sample application suite demonstrates a number of COM and distributed application features in Visual Basic 5.0. It also models distributed application performance characteristics. Detailed information about all the components in this sample suite can be found in Building Client/Server Applications With Visual Basic.

## AEClient

File	Description
modCInt.bas	Client main/global utility module
clsPooTl.cls	Client class for pooled tests
clsQueTl.cls	Client class for queued tests
clsCalbk.cls	Client class for callback notification
clsCntSv.cls	Client class for service request callback information
clsDrtTl.cls	Client class for direct instantiation tests
Client.cls	Client class
frmCInt.frm	Client main form
frmCInt.frx	Client main form binary information
client.ico	Client form icon
AEClient.rc	String resources source file
AEClient.res	String resource file
AEClient.vbp	Client component project file

## AEExpdtr

File	Description
modExpdt.bas	Expeditor main/global utility module
SyncRtn.cls	Expeditor notification class for client event source
CallBkRf.cls	Expeditor result information class
Expeditr.cls	Class of main expeditor interface
frmExpdt.frm	Expeditor main form
frmexpdt.frx	Expeditor main form binary information
expetitr.ico	Expeditor main form icon
AEExpdtr.rc	String resources source file
AEExpdtr.res	String resource file
AEExpdtr.vbp	Expeditor component project file

## AEInstnr

File	Description
modInstr.bas	Process instancer main/global utility module
Instnccr.cls	Process instancer main class
AEInstnr.vbp	Process instancer component project file

## AELogger

File	Description
------	-------------

modLoggr.bas	Logger main/global utility module
Logger.cls	Logger primary class
frmLoggr.frm	Logger main form
frmloggr.frx	Logger main form binary information
logger.ico	Logger main form icon
AELogger.rc	String resources source file
AELogger.res	String resource file
AELogger.vbp	String component project file

## **AEPool**

<b>File</b>	<b>Description</b>
modPool.bas	Pool manager main/global utility module
PoolMgr.cls	Pool manager main class
Pool.cls	Pool manager client interface class
frmPool.frm	Pool manager main form
frmpool.frx	Pool manager main form binary information
pool.ico	Pool manager main form icon
AEPool.rc	String resources source file
AEPool.res	String resource file
AEPool.vbp	Pool manager component project file

## **AEQueue**

<b>File</b>	<b>Description</b>
modQueue.bas	Queue manager main/global utility module
Queue.cls	Queue client interface class
clsQueDI.cls	Queue handoff/polling class
clsServc.cls	Queue service request class
QueueMgr.cls	Queue manager primary class
frmQueue.frm	Queue manager main form
frmQueue.frx	Queue manager main form binary information
qmanager.ico	Queue manager main form icon
AEQueue.rc	String resources source file
AEQueue.res	String resource file
AEQueue.vbp	Queue manager component project file

## **AEServic**

<b>File</b>	<b>Description</b>
modServc.bas	Service main/global utility module
Service.cls	Service class implementation
AEServic.mdb	Service sample data file
AEServic.rc	String resources source file
AEServic.res	String resource file
AEServic.vbp	Service component project file

## **AEWorker**

<b>File</b>	<b>Description</b>
-------------	--------------------

modWorkr.bas	Worker main/global utility module
Worker.cls	Worker class implementation
AEWorker.rc	String resources source file
AEWorker.res	String resource file
AEWorker.vbp	Worker component project file

### **AEWrkPvd**

<b>File</b>	<b>Description</b>
basWrkPd.bas	Worker provider main/global utility module
clsWrkPd.cls	Worker provider class for worker threads on remote machines
AEWrkPvd.rc	String resources source file
AEWrkPvd.res	String resource file
AEWrkPvd.vbp	Worker provider component project file

### **AEIntrfc**

<b>File</b>	<b>Description</b>
AEIntrfc.ODL	COM Interface specification for the Worker extensibility interface

### **Include**

<b>File</b>	<b>Description</b>
modAECon.bas	Global APE constants
modWinEr.bas	Global Win32 error constants
modVBErr.bas	Global VB error constants
modAEGlb.bas	Global APE utility module
clsWkMac.cls	Worker remote machine information class
clsPosFm.cls	Form position/persistence class
clsWorkr.cls	Worker instance information class

### **Server**

<b>File</b>	<b>Description</b>
AEExpdtr.cmp	Expeditor compatible server implementation
AEInstnr.cmp	Process instancer compatible server implementation
AELogger.cmp	Logger compatible server implementation
AEClient.cmp	Client compatible server implementation
AEPool.cmp	Pool manager compatible server implementation
AEQueue.cmp	Queue manager compatible server implementation
AEServic.cmp	Service compatible server implementation
AEWorker.cmp	Worker compatible server implementation
AEWrkPvd.cmp	Worker provider compatible server implementation

**Note** APE can be run entirely on one machine, or various components can be distributed to remote machines using Remote Automation or DCOM to benefit from distributed processing power and support for multi-user access. APE does not have to be altered to take advantage of this -- it is completely configurable.

