Microsoft Visual Basic Version 5.0

Evaluators Guide

The most efficient tool for creating high-performance Windows®, client/server and Web-based applications and components.

March 1997

Contents

Continued on next page

Continued from previous page

Introduction

Six years ago, Microsoft Corporation launched a revolution in the way developers create applications. For the first time, programmers could quickly develop Windows-based applications using the Microsoft® Visual Basic® programming system. Having been mastered by more than 3 million developers, Visual Basic has proven to be the most popular tool for creating high-performance components and component-based applications.

With the introduction of the Visual Basic programming system version 5.0, Microsoft launches a revolution of greater dimension, offering programmers the first rapid application development (RAD) tool to merge client/server and Internet technologies. No longer will developers be required to choose between performance and productivity, application scalability and a development tool that is easy to use, or local area network (LAN) and Internet deployment models. With Visual Basic version 5.0, developers gain:

- **Performance and scalability.** Advanced features such as optimized native code compilation, enhanced database access, and the Microsoft® Transaction Server provide state-of-the art application performance and scalability for traditional client/server, three-tier distributed, and Internet application architectures.
- **Productivity.** A radically enhanced Integrated Development Environment (IDE) with IntelliSense™, ActiveX™ Control creation capabilities, rich component encapsulation and reuse, an industry-leading array of third-party components, and the ability to use existing Visual Basic skills, code, and technology. Offers developers unprecedented levels of productivity for application design, development, and deployment.
- **Internet options.** The ability to easily repurpose existing client/server applications to meet business needs and opportunities of Internet technologies while maintaining a common component-based design for nearly all project elements.

All of this means that with Visual Basic version 5.0, developers can build production-quality applications within budget and schedule constraints while writing less code.

Top new features of Visual Basic version 5.0 include:

Productivity	Performance	Data Access	Internet
Enhanced IDE with IntelliSense™ and an extensive set of wizards	Native code with faster loading forms and controls	Microsoft SQL Server 6.5	Browser-based application development and Migration Wizard
Visual ActiveX Control creation	Microsoft Transaction Server	Enhanced RDO 2.0 with UserConnection Designer	Visual ActiveX Control creation
Enhanced controls and tools	Integrated DCOM support	Inline SQL Server stored procedure debugging	Setup Wizard for Internet deployment
IDE extensibility for smarter add-ins and customization	Application and database performance tools	Enhanced Oracle and Microsoft SQL Server drivers	Web browser, HTTP/FTP, and Winsock Internet controls

As you read this evaluators guide, you will see how Visual Basic implements innovations in software application development while offering developers options to use existing code, designs, and skills to more quickly build and deploy faster applications.

This guide consists of the following sections:

- Evaluation criteria for rapid application design (RAD) client/server application development tools
- An overview that demonstrates how version 5.0 meets the real-world needs of developers.
- A description of the core features of Visual Basic version 5.0
- A description of the enterprise-oriented features of Visual Basic version 5.0.
- A test drive of Visual Basic version 5.0.

Evaluating RAD Client/Server Application Development Tools

Customer research reveals that there are 10 separate categories programmers use when evaluating rapid application development (RAD) client/server development tools. Client/server RAD tools must meet these criteria in order to be considered complete. We suggest that each area be weighted according to this table.

Details about each area follow this table.

Criterion	**Percentage of Total Score**
High performance	15%
Component development and reuse	15%
Simplified data access and retrieval tools	15%
Scalability	10%
Database design and modeling tools	10%
Internet integration	10%
Programmer productivity	10%
Client/server debugging	5%
Application deployment flexibility	5%
Application and development tool extensibility	5%
Total Score	**100%**

High Performance

- **Fast server-based data access.** Make data access as fast as direct ODBC API coding.
- **Deliver C++-level performance.**
- **Take advantage of Windows NT®.** Provide all of the benefits of multi-threading without multi-threading programming. Provide native-code applications for platforms that run the Windows® 95 and Windows NT® operating systems, including Intel, and RISC. Run the same code and support unmodified code recompile across each of these platforms.

Component Development and Reuse

- **Simplify use and management of components.** Components created with other languages must be simple to integrate. The functionality contained within a new component must be easy to view. The programming tool must allow easy identification of different versions of the same component. The tool must protect a programmer from creating incompatible new versions of existing components.
- **Simplify component creation.** Support new components being created by assembling existing components much the same way applications are created, by modifying existing components, or components can be created from scratch if necessary.

Simplified Data Access and Retrieval Tools

- **Automatic code generation.** Automate the creation of each tier within a client/server application.
- **Simplify access to stored procedures.** Make stored procedure calls intrinsic to the language.
- **Provide data-tier debugging.** Make remote stored procedure calls like any other call in the programming language.
- **Minimize user waiting.** Support the strategies and technologies that minimize user delays caused by time-consuming remote operations, such as establishing connections, slow network performance, and heavy resource use.

Scalability

- **Environment.** Offer a scalable environment for servers and business rules.
- **Reliable and safe environment.** Offer a reliable and safe environment for executing distributed transactions. Support components being used with the integrity and rollback capabilities normally associated with database transactions. Move the complexity of managing and scheduling resources from applications to the environment.
- **Manage the environment.** Handle the complexity involved by creating reliable server applications.

Database Design and Modeling Tools

- **Simplify creation of complex queries.**
- **Include database schema design tools.** Enable and automate database creation.
- **Enable the prototyping of server-side stored procedures.**
- **Simplify the prototyping of distributed applications.** Support what-if performance analysis based on users, network traffic, and application design. Provide automated stress testing.
- **Enable local RDBMS development.**

Internet Integration

- **Deliver and install applications using the Internet.** Existing applications must function and behave more like a Web page. In other words, support delivering and installing them by typing a URL.
- **Enable existing applications to access the Internet.** Integrating existing applications with Web sites, transferring and receiving files using FTP, and integrating other popular Internet protocols.
- **Enable Internet access for existing applications.** Existing client/server components must be accessible on a LAN and over the Internet.

Programmer Productivity

- **Simplify or reduce coding.** Produce or automate much of the mundane code that must always be written. Deliver this in the form of templates, wizards, or galleries.
- **Make learning and using new components easier.** Anticipate the needs of a programmer writing code. Eliminate the need to memorize correct syntax and proper usage of new components.
- **Enable full customization of the development environment.** Programmers, unlike typical users, need to continually refine their tools. Customization of the programmer user interface and creation of new tool features should be accomplished with and without programming.

- **Manage source code for the team.** Typically, teams of programmers build client/server applications. Collaboration and code sharing must be natural, well-integrated, and out of the way.

Client/Server Debugging

- **Enable client/server application debugging.** Support debugging of all parts of a distributed application. Debugging should start in the application and then step into and out of all components, whether built using native or mixed languages.
- **Support interactive debugging into and out of remote stored procedure calls.** Support this level of debugging on the Relational Database Management System.

Application Deployment Flexibility

- **Support the development of multi-tier applications**. Individual tiers must execute either locally or on remote systems. All tiers must be accessible across a WAN and over the Internet.
- **Support Windows NT DCOM.** Accessing remote components must be transparent to a programmer.

Application and Development Tool Extensibility

- **Support integration of commercial component libraries.** Support application creation with full support for integrating component libraries.
- **Support integration of third-party tools.** In the development environment, support integration of third-party tools such as CASE, testing, bug tracking, modeling, source-code control, and so forth.
- **Support in-house tool customization.** Offer programmers the option to customize tools.

Overview: 1997 Application Development Requirements and Visual Basic Version 5.0

A well-designed application is one that is usable, flexible, resistant to failure, testable, and deployable—and one that can also be built on schedule and within budget. It takes a comprehensive development tool to give developers the capabilities they need to meet these criteria. Scalability is also important—as an intrinsic trait, not as an afterthought. Additionally, the development tool must be open, offering a choice of implementation architectures, integration of disparate systems, and component reuse. In the following sections, you will see how Visual Basic version 5.0 meets each of these requirements.

Comprehensive

Visual Basic version 5.0 is a comprehensive tool both because of the ways different types of users can use it to be productive and in the deployment flexibility it offers.

Productivity

Using a productive development tool, people with differing backgrounds and skill levels can rapidly become proficient at building solutions. This requirement is not a one-time issue, but one that affects all companies, if for no other reason than employee turnover. A productive tool also supports migration by the development team from host-system development, so the team can learn by applying experience acquired on other platforms. At the same time, it avoids obstructing experienced developers with features designed for the beginner. Finally, a productive development tool abstracts developers from platform idiosyncrasies. True, developers must deal with databases, networks, and multi-tier architectures, but they should not need to know the underlying details of each in order to build an application.

Key capabilities of Visual Basic version 5.0 that enhance developer productivity include:

- **Refined development environment.** The new multiple-document interface, with its docking and linking windows, eliminates many unnecessary mouse and keyboard motions and allows developers to organize their development environment in whatever way is most productive for them.

- **IntelliSense™.** Features such as Quick Info, Data Tips, List Members, and List Constants eliminate the need to memorize syntax, search for parameters, and learn object models. Visual Basic offers developers a list of everything known to the environment that can logically be typed at a particular point.
- **Flexible execution model.** Visual Basic version 5.0 is the only development tool that gives a developer the productivity of an in-process interactive debugger combined with the performance of native compiled code.
- **Object-oriented language improvements.** Version 5.0 helps developers increase code reuse through interface inheritance—allowing classes to reuse interfaces from other component object model (COM) classes. Other language improvements simplify or enhance the benefits of abstraction, encapsulation, and polymorphism.
- **Sophisticated add-ins.** Visual Basic version 5.0 comes with a new breed of wizards and builders that fully use the enhanced IDE object model. As a result, developers can visually build and manipulate class modules, properties, events, and methods. They can also visually define ActiveX Control interfaces, design property pages, and extrude sub-classed control members, that is, create an interface that delegates responsibilities to the respective controls.
- **User connection designer.** Developers can visually design queries and then reference them as objects, allowing stored procedures to be referenced with the same syntax and IntelliSense features as ActiveX Components.
- **SQL Server debugging.** Using T-SQL debugger, developers can interactively debug Transact SQL—Microsoft SQL Server™ stored procedures—at the same time as they develop and debug Visual Basic–based applications—and on the same workstation.

Deployment flexibility

Resolving issues associated with application deployment can consume more resources than building the application. The deployment methodology represents the single best area for shortening project implementation duration, decreasing costs, and reducing ongoing support burdens.

Over time, organizations have learned that overall productivity can be increased and resource costs decreased by providing employees with the right tools for the work at hand. A desktop computer with both generic and business-specific software meets the broadest set of needs. Still, installing and updating that software takes time and costs money. Here's where the Internet appeals: a user need only load the browser to install and use applications. At the same time, certain business requirements can still be better addressed with client/server technologies.

Visual Basic version 5.0 introduces Active Documents, a new technology that bridges the gap between client/server, Internet, and intranet technologies—and leverages the best of all three. Active Documents run in the Internet browser. When the user clicks a hyperlink, the associated application is automatically installed on the user's desktop. With this integration between the capabilities of different technologies, a developer can:

- Continue to build applications the same familiar way.
- Use the software development advantages of the client/server-computing paradigm.
- Automatically install and update applications using the same pull download model as used for ActiveX Controls.
- Use hyperlinks to link HTML-based applications with Visual Basic–based applications.

Scalable

A scalable tool must take into account both design-time and run-time scalability. Any discussion of scalability must also consider application performance. This section explains how version 5.0 addresses all three scalability issues.

Design-time scalability

Most people define scalability as an application's ability to adapt to meet the changing needs of the business including changes in the number of users, data, and transactions. Before an application can scale, however, it must be built—and the development tool used to build it must also be scalable and capable of supporting any size team, from a single developer to a large group. This is what is meant by design-time scalability.

In addition to the multi-tier application architecture introduced in Visual Basic version 4.0, Visual Basic version 5.0 has several new design-time scalability features:

- **SQL Server, Developer Edition.** The Enterprise Edition of Visual Basic version 5.0 includes Microsoft SQL Server, Developer Edition, so developers can develop, debug, and test applications combining SQL Server and Visual Basic on either a workstation or workgroup server.
- **Multiple projects in one session.** Using the new development environment, a developer can load several projects into a single Visual Basic session. Using this capability, programmers can develop, debug, and compile three-tiered architectures and ActiveX Controls—regardless of their component distribution architecture—as if they were a single, monolithic application.
- **Reusable code through ActiveX Controls.** Developers can build common user interfaces once as an ActiveX Control and then use them in many places throughout the project, reducing the sharing requirements of a single form.

Run-time scalability

Simply put, run-time scalability is the capability of a development tool to create applications that can grow or shrink to support changes in the number of users, size of databases, and number of transactions. Run-time scalability was not a problem in the early days of rapid application development because client/server applications did not generally require multi-gigabyte databases, nor did they target extremely large numbers of users. Today, however, client/server applications are being deployed to all kinds of desktops, and databases are storing information from around the world, in addition to information from the department.

Enhancements in Visual Basic version 5.0 not only make it possible to create scalable distributed applications, but make the process easier than ever. Key capabilities include:

- **Microsoft Transaction Server (formerly known as “Viper”).** This tool gives developers the power and flexibility of a transaction processing monitor, object broker, and transaction server without the typical complexity and overhead.
- **Multi-threading and thread safety.** With version 5.0, developers can use the unattended execution feature to create multi-threaded and thread-safe ActiveX Components for use in three-tiered applications and Active Server Pages. ActiveX Components marked for unattended execution require significantly less memory because they abide by the COM apartment-threading model.

- **Remote Data Objects (RDO) version 2.0.** RDO 2.0 includes new features such as client-side cursors, batch updates, and an event model to allow more flexible processing as well as asynchronous and concurrent application architectures.
- **Native Alpha code.** Using version 5.0, developers can use the power of RISC processor architectures with native compiled code for Alpha processors.

Application performance

The number one complaint about many new applications is unacceptable performance. Creating applications with acceptable performance is possible, but doing it within schedule and budget is a challenge. Visual Basic version 5.0 has several new features that address both of these performance issues:

- **Native code compiler.** The native code compiler improves application performance, as much as 20 times faster, by using the Microsoft Visual C++® optimizing compiler technology.
- **ActiveX Control creation.** Controls included with version 5.0 are based on ActiveX technologies, making them on average up to three times faster than version 4.0 controls.
- **Accelerated forms.** Forms processing has been tuned to display forms up to 17 times faster.
- **Remote Data Objects (RDO) version 2.0.** RDO 2.0 is up to 46 percent faster than RDO 1.0, and it includes new features such as client-side cursors and batch updates that use asynchronous processing to eliminate unnecessary delays.
- **Performance details.** See Appendix A: Client/Server Development Tool Performance Analysis, by Carnegie Technologies, for details about the performance improvements of Visual Basic version 5.0.

Open

Microsoft's overall goal regarding openness is to support standards broadly adopted by our customers in order to offer developers choice, flexibility, and the ability to create solutions capable of targeting a variety of clients. Additionally, to help our customers protect their investment in existing software while also taking advantage of the latest technology developments, we design our products to be inter-operable with both existing legacy systems and emerging business solutions.

Visual Basic version 5.0 offers three types of openness: the ability to work with and connect to a broad range of products and solutions; the flexibility to tailor and extend the work environment with other languages and tools that meet additional needs; and the ability to easily build components for a variety of clients.

Broad inter-operability and connectivity

Aspects of Visual Basic version 5.0 that make it easy for Visual Basic to work with and connect to a broad range of products and solutions include:

- **Open standards.** All applications, controls, and components created in Visual Basic version 5.0 conform to ActiveX, COM, and DCOM standards, so a developer can select the application partitioning and component distribution architectures that best meet the needs of an application. Additionally, controls and components created in version 5.0 can be used not only in Visual Basic applications, but also in Microsoft® Office 97 applications, any of the more than 100 licensed Visual Basic, Applications Edition products, Visual FoxPro™ version 5.0 applications, and Internet applications.
- **Buy-before-build choice.** Visual Basic 5.0 conforms to ActiveX technologies, offering developers access to the largest library of reusable components available. In its first release, developers could choose from numerous off-the-shelf, generic components that easily integrated with solutions that they built themselves. Now that version 5.0 makes it so easy to create controls and add-ins, developers will be able to choose from a wide range of vertical as well as generic extensions, and build their own custom ActiveX Controls.
- **Connectivity standards.** Using Visual Basic, developers have always been able to build solutions that co-exist and inter-operate with existing production systems. Previous versions also standardized on ODBC, allowing Visual Basic applications to connect to the widest range base of database management systems possible. Version 5.0 further enhances this connectivity and interoperability with its component development approach and use of COM-based technologies such as the technology currently code-named “Cedar,” a

technology being designed to connect Visual Basic solutions to mainframe-based transaction processing systems.

Flexibility

In a typical project, user participation starts out strong, with developers intending to produce a very user-centric product. However, user participation generally diminishes as a project moves from conceptual to physical design and construction, and as developers find that their tool doesn't support building what the users want. With ever-present schedule and cost constraints, developers generally choose the easiest approach, resulting in less emphasis on user input and diminished user participation.

Visual Basic version 5.0 avoids these problems by offering developers extensive flexibility. Aspects of the product that promote flexibility include:

- **Customizable development environment.** The Visual Basic version 5.0 development environment is completely customizable. Developers can dock and link Windows and toolbars in any manner they want to suit their work style, as well as can configure third-party add-ins and custom forms for docking and linking. They can even opt not to use the new multiple document interface and stick with the single document interface of previous versions.
- **Extensibility.** Using custom ActiveX Controls and add-ins, developers can extend the Visual Basic development environment to meet specific development requirements, to comply with industry-specific requirements, or to provide additional productivity or customization tools.
- **Ability to use the same code in client/server and Web applications.** With version 5.0, component-based business functions developed in earlier versions of Visual Basic can now be used in Web and browser-based applications.

Cross-platform capabilities

Visual Basic plays an important role in cross-platform applications. However, unlike Microsoft Visual InterDev™, cross-platform is not the primary mission of Visual Basic. Examples of cross-platform capabilities in Visual Basic version 5.0 include:

- **Active Server Pages.** Components developed with Visual Basic can be used as ISAPI or Active Server Pages, making their functionality accessible by all browsers, whether running with Windows, Apple Macintosh, or UNIX operating systems.

- **Native Alpha code.** Using version 5.0, developers can use the power of RISC processor architectures with native compiled code for Alpha applications and components.
- **ODBC data access.** Visual Basic completely supports ODBC, making ODBC databases on any platform accessible through common, object-oriented data access mechanisms.

New Visual Basic Version 5.0 Features

Key features of Visual Basic version 5.0 include:

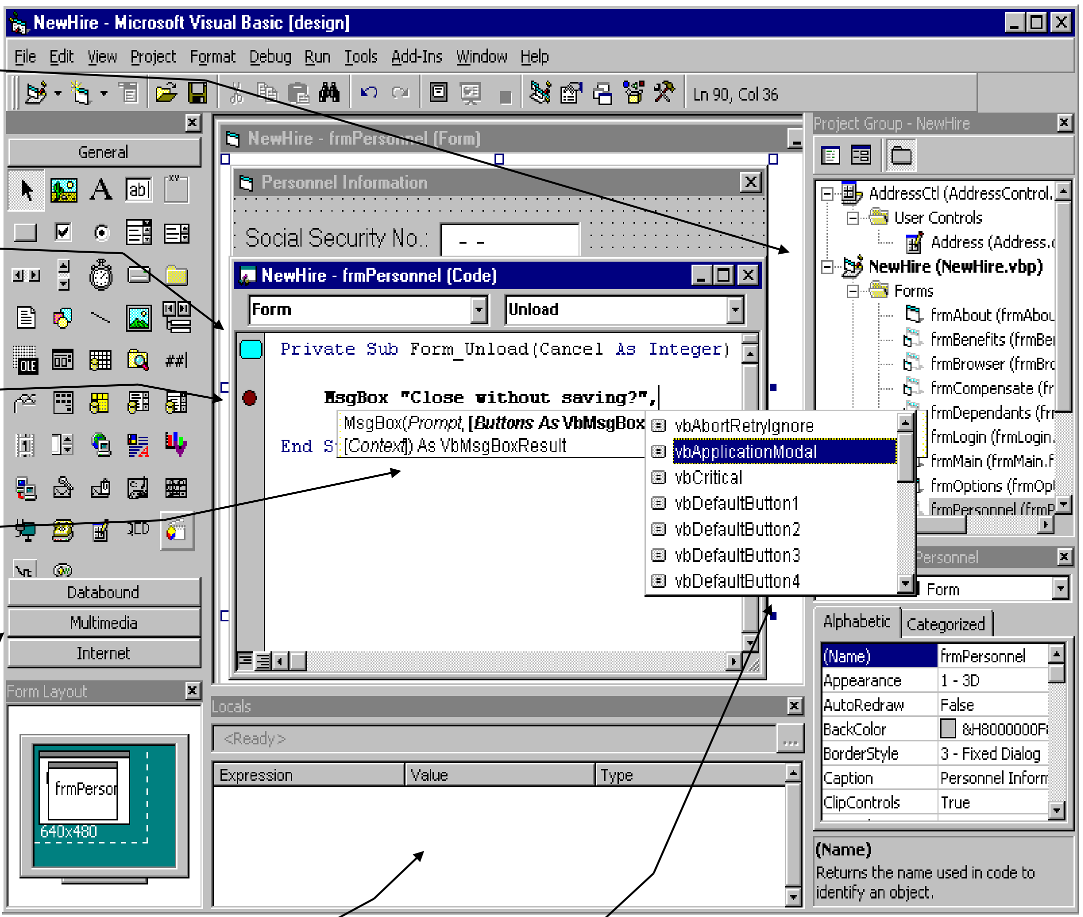
- Enhanced integrated development environment.
- Native code compiler.
- Language improvements.
- Internet features.
- Enhanced extensibility.
- Broad set of enterprise features.

The following sections provide details on each of these features. A section is also dedicated to the array of enterprise features.

Enhanced Integrated Development Environment

Much of the popularity of Visual Basic is a result of the integrated development environment (IDE), which significantly simplifies the task of creating applications. The Visual Basic version 5.0 IDE, illustrated in Figure 1, offers new capabilities. The complex has been simplified, the repetition has been automated, and a dose of intelligence has been added. The IDE is a common development environment consisting of a forms package, a powerful code editor, an object browser, and a debugger. All editions of Visual Basic version 5.0 share this common development environment—as do all products that incorporate Visual Basic, Applications Edition, including Microsoft Excel, Microsoft Access, Microsoft Word, Microsoft PowerPoint®, and Microsoft Project. Together, the Visual Basic language and Visual Basic, Applications Edition, represent the most-used RAD programming system.

In response to the demand for the Visual Basic programming model, Microsoft recently announced a licensing program for Visual Basic, Applications Edition, so other companies can incorporate it into their products. As a result, developers who use Visual Basic can now program not only with Microsoft products, but also with software written by companies such as Adobe, AutoDesk, Visio, and SAP.



```
Private Sub Form_Unload(Cancel As Integer)

    MsgBox "Close without saving?",
End S
```

MsgBox(*Prompt*, **[*Buttons* As VbMsgBox**
[*Context*]) As VbMsgBoxResult

vbAbortRetryIgnore
vbApplicationModal
vbCritical
vbDefaultButton1
vbDefaultButton2
vbDefaultButton3
vbDefaultButton4

(Name)	frmPersonnel
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F
BorderStyle	3 - Fixed Dialog
Caption	Personnel Inform
ClipControls	True

Load and debug ActiveX Controls, ActiveX DLLs, and executable files at the same time.

Locate specific code quickly with bookmarks.

Toggle Breakpoints with a mouse click.

View syntax and parameter information for Visual Basic statements and custom functions with Quick Info.

Manage the growing number of controls with Toolbox tabs.

Set form startup position according to various screen resolutions with the Form Layout window.

Debug all declared variables which automatically show in the Locals window.

Select from a list of valid constants for statements and custom functions with List Constants.

Manipulate properties alphabetically or categorically and view property description in the Property window.

Figure 1. The integrated development environment (IDE) in version 5.0 relieves experienced developers of repetitive tasks and shortens the learning curve for new developers.

Multiple projects

Visual Basic® version 5.0, like Visual C++®, Visual J++™, and Visual InterDev™, offers the ability to work on several projects at once. This capability is extremely useful for designing and debugging reusable ActiveX Controls and three-tier applications. With several projects loaded simultaneously, a developer can debug multiple projects in the same manner and with the same ease as single projects. For example, by setting breakpoints in the source code of an application, a developer can debug it line by line, starting in the host project, stepping into the component projects as they are called, and moving back out to the host again. Without it, application development can be slowed. Previously, developers loaded multiple instances of Visual Basic for each component and configured the instances to work properly—a time-consuming approach.

The Project Explorer also makes it easier to work with multiple projects at one time because it displays—in an outline view—components for all loaded projects associated with each currently open project. For example, it might display all associated ActiveX Controls, forms, classes, modules, resource files, property pages, and Active Documents. Each project appears as a new root in the outline. To easily manage a workspace with several projects open, a developer can expand or collapse a folder to show or hide all open components—forms, code modules, and so forth.

Enhanced Properties Window

A developer can use the enhanced Properties window to display properties of UserControl objects, forms, modules, classes, property pages, and Active Documents.

The Properties window has two tabs: the Alphabetic view familiar from previous versions and a new Categorized view. The Alphabetic view displays an alphabetical list of properties. The Categorized view groups properties by type—for example, all properties related to color, font, or position. Developers can expand or collapse categories in the Properties window.

Developers can also specify a category to expose for any custom properties of an ActiveX Control they create using Visual Basic version 5.0. Since ActiveX Components are based on COM, they all contain type information to describe their interfaces. Visual Basic makes extensive use of this information, easily determining all exposed objects, properties, methods, and even version numbers of ActiveX Components. The Property window also works in this way, displaying the relevant type information whenever a developer selects a property.

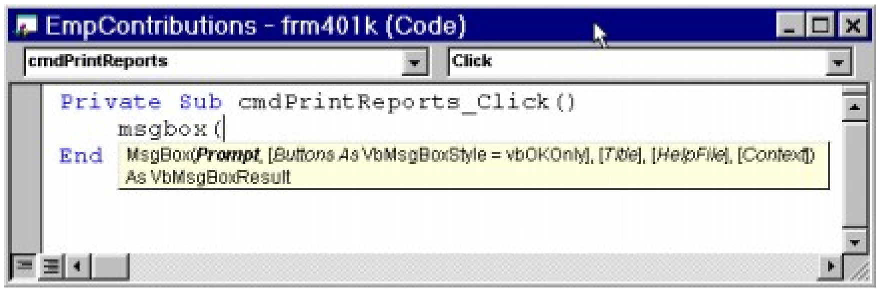
Code editor intelligence

Every few years, a surprising feature innovation occurs. The best innovations seem to provoke “Why didn’t I think of that?” Examples include Microsoft® Word IntelliSense features like AutoCorrect, AutoSpellcheck, and drag-and-drop editing. Visual Basic 5.0 brings IntelliSense to developers. With previous versions of Visual

Basic, developers needed a thorough understanding of both language syntax and COM. IntelliSense minimizes this requirement by offering all the information that developers need when they need it, making it easier and faster to write error-free code.

IntelliSense works with Visual Basic, third-party, and user-created ActiveX Components. Key IntelliSense features include:

- **Quick Info,** shown in Figure 2, gives a developer syntax assistance for statements, procedure calls, methods, and properties.

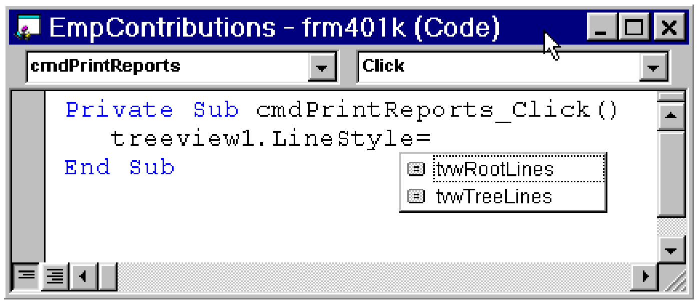


```
Private Sub cmdPrintReports_Click()
    msgbox(
End  MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context])
     As VbMsgBoxResult
```

Figure 2. Quick Info offers developers syntax assistance.

- **Complete Word** completes the word being typed once a developer types enough letters to make it distinct. For example, in the code window, when a developer types `msg` followed by CTRL+SPACEBAR, the word `MsgBox` is completed.

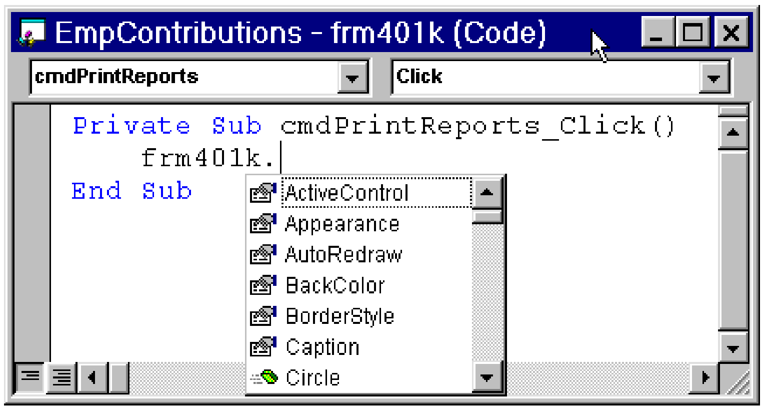
- **List Constants**, illustrated in Figure 3, displays a list box with constants that are valid choices for a property.



```
Private Sub cmdPrintReports_Click()
    treeview1.LineStyle=
End Sub
```

Figure 3. List Contents displays a list box showing the constants that are valid choices for a property.

- **List Properties/Methods,** illustrated in Figure 4, displays a list box containing the properties and methods available for the object.



```
Private Sub cmdPrintReports_Click()
    frm401k.
End Sub
```

Figure 4. The List Properties/Methods feature displays a list box showing the properties and methods available for the object.

Dockable and linkable windows

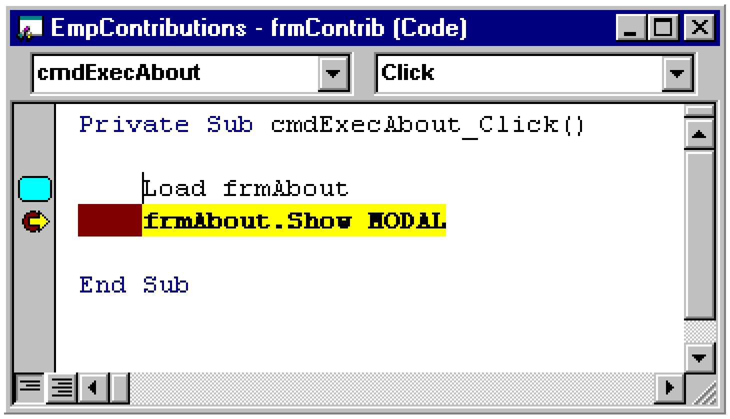
Dockable and linkable windows are new features that make it easier to manage and organize multiple windows. With the docking feature, when a developer places a child window—such as a toolbar or Toolbox—near the outer edge of the Visual Basic parent window, the child window attaches itself to the parent window. With the linking feature, when any two child windows are placed close together, they link and become one.

New debugging features

Along with features to improve the speed and ease of creating applications, developers are always looking for ways to save time in the debugging process. To assist them, we've added new features specifically aimed at improving debugging productivity—and we've made existing features more accessible. Key debugging enhancements include:

- **Cross-project debugging.** Multiple Visual Basic projects can be debugged in one instance of Visual Basic using the new multiple projects feature.
- **T-SQL debugging.** For the first time, developers can debug Microsoft SQL Server stored procedures with the same ease with which they debug Visual Basic code. You can read more details in the Enterprise Features section.
- **Margin indicators.** This feature, illustrated in Figure 5, offers visual cues for breakpoints, current line of execution, bookmarks, and CallStack markers. Developers can also run to a specific line, set a debug statement by simply

clicking it, set the next statement to run by dragging the current-line indicator, and set bookmarks for quick access to specific areas of code.

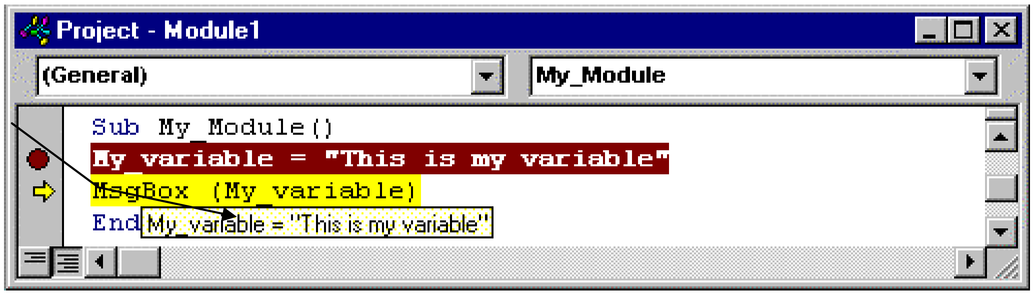


```
Private Sub cmdExecAbout_Click()

    Load frmAbout
    frmAbout.Show MODAL

End Sub
```

Figure 5. Margin indicators offer visual cues and drag and drop capability for breakpoints, next statement, and bookmarks.

- **DataTips.** When Visual Basic is in break mode and a developer positions the cursor over a variable, an informational window, like a ToolTip, displays the value of the variable, as seen in Figure 6.

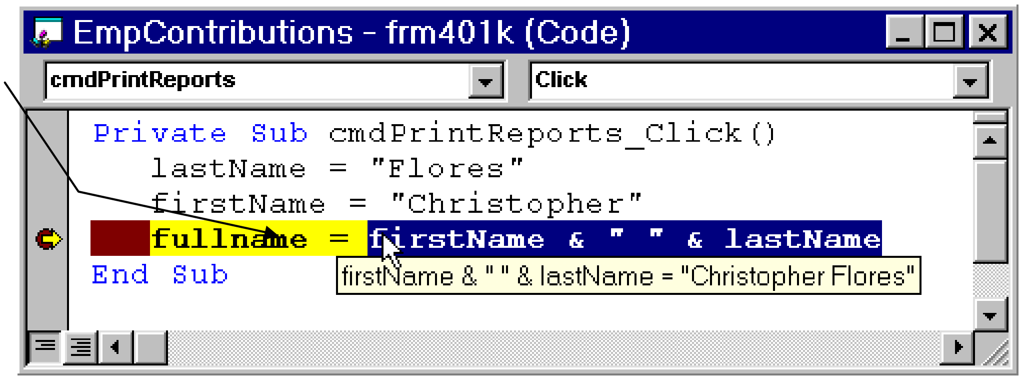
Data Tips display the value of variable when the cursor hovers over it.



```
Sub My_Module()
My_variable = "This is my variable"
MsgBox (My_variable)
End
```

My_variable = "This is my variable"

Data Tips display the value of partially selected expressions.

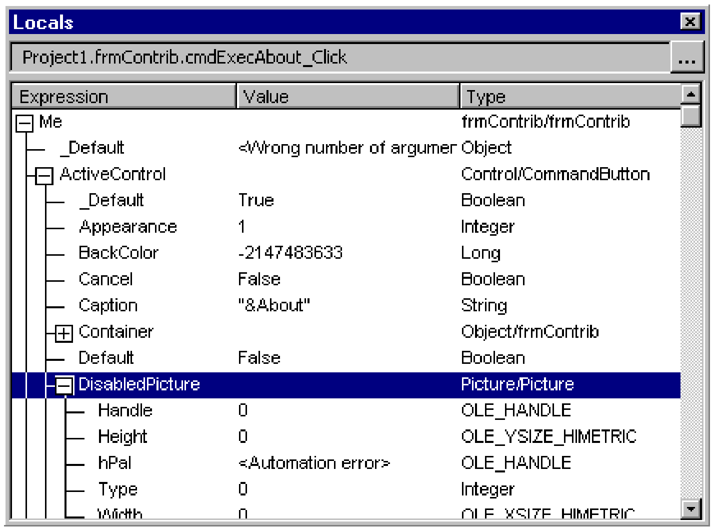


```
Private Sub cmdPrintReports_Click()
    lastName = "Flores"
    firstName = "Christopher"
    fullname = firstName & " " & lastName
End Sub
```

firstName & " " & lastName = "Christopher Flores"

Figure 6. Data Tips offers a developer helpful information about variables and expressions.

- **Locals window.** The Locals window, shown in Figure 7, automatically displays all the declared variables in the current procedure and the value of each variable. Additionally, a developer can change values of variables in the Locals window.



Project1.frmContrib.cmdExecAbout_Click

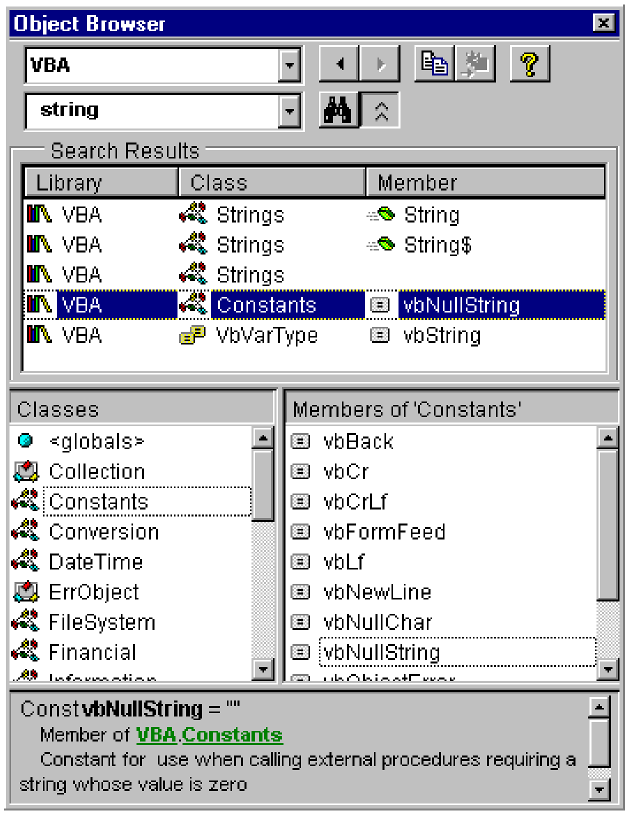
Expression	Value	Type
Me		frmContrib/frmContrib
_Default	<Wrong number of argumer	Object
ActiveControl		Control/CommandButton
_Default	True	Boolean
Appearance	1	Integer
BackColor	-2147483633	Long
Cancel	False	Boolean
Caption	"&About"	String
Container		Object/frmContrib
Default	False	Boolean
DisabledPicture		Picture/Picture
Handle	0	OLE_HANDLE
Height	0	OLE_YSIZE_HIMETRIC
hPal	<Automation error>	OLE_HANDLE
Type	0	Integer
Width	0	OLE_XSIZE_HIMETRIC

Figure 7. The Locals window automatically displays all the declared variables in the current procedure, and is used by a developer to change the value.

- **Watch window.** Version 5.0 makes it easier to use the Watch window. Instead of typing a variable in the Watch window, a developer can now just drag a variable into it from the code window rather than specifying everything to watch. Like the Locals window, a developer can change values in the Watch window.

- **Enhanced immediate window.** Using the Immediate window, now a developer can run code in it at design time and in break mode.

- **Enhanced object browser.** With the enhanced Object Browser, shown in Figure 8, a developer can search for a class or member in a specific library or across all referenced libraries and easily jump to the References dialog box or to modules and procedures in projects. The new Description pane reminds them about object use.



VBA

string

Library	Class	Member
VBA	Strings	String
VBA	Strings	String$
VBA	Strings	
VBA	Constants	vbNullString
VBA	VbVarType	vbString

Classes	Members of 'Constants'
<globals>	vbBack
Collection	vbCr
Constants	vbCrLf
Conversion	vbFormFeed
DateTime	vbLf
ErrObject	vbNewLine
FileSystem	vbNullChar
Financial	vbNullString

Const **vbNullString** = ""
Member of VBA.Constants
Constant for use when calling external procedures requiring a string whose value is zero

Figure 8. The Object Browser provides easy access to project modules and procedures.

Enhanced editor

The enhanced editor increases developer productivity and code readability with features such as syntax checking, color-coded syntax, block indent and outdent, block comment and uncomment, several frequently requested navigation keys, and word-processor-like drag-and-drop editing within and across code windows.

Forms

Historically, one of the most productive features of Visual Basic has been the visual forms design environment, which can save hours of coding. Aligning and formatting controls, however, took time away from writing code. To solve this problem, third parties have taken advantage of the extensibility of Visual Basic to create add-ins that assist with aligning forms.

In version 5.0, productivity in designing forms will improve still further with the addition of advanced alignment capabilities usually found only in sophisticated drawing packages. Using these new capabilities, a developer can select one or more controls on a form and manipulate them as if they were objects. The new capabilities include:

- **Align,** which moves selected objects to left, center, right, top, middle, bottom, or to grid.
- **Make Same Size,** which resizes selected objects to the size of the last object selected in width, height, or both.
- **Size to Grid**, which formats a selected object to fit gridlines in the form.
- **Horizontal Spacing,** which changes the horizontal space between selected objects. Spacing options include Make Equal, Increase, Decrease, or Remove spacing.
- **Vertical Spacing,** which changes the vertical space between selected objects, providing the same options as the Horizontal Spacing command.
- **Center in Form,** which centers selected objects on the central axes of the form.
- **Order,** which changes the sequence of selected objects on a form, using the Bring to Front or Send to Back command.

A new form-layout window further simplifies form design. A developer can drag and drop the form in the window, or use the new StartPosition Property to set the startup position.

Native Code Compiler

In the past, building high-performance applications sometimes meant sacrificing developer productivity and working with tools that were difficult to learn and use. Features in Visual Basic version 5.0 make those sacrifices unnecessary. By combining the world-class, optimizing compiler technology built by Microsoft with other performance improvements listed below, Visual Basic version 5.0 produces native code applications that run up to 20 times faster than applications built with earlier versions of Visual Basic.

The CPU-intense benchmarks illustrated in Figure 9 show that for compute-intensive operations, Visual Basic version 5.0 is 20 to 60 times faster than Visual Basic version 4.0.

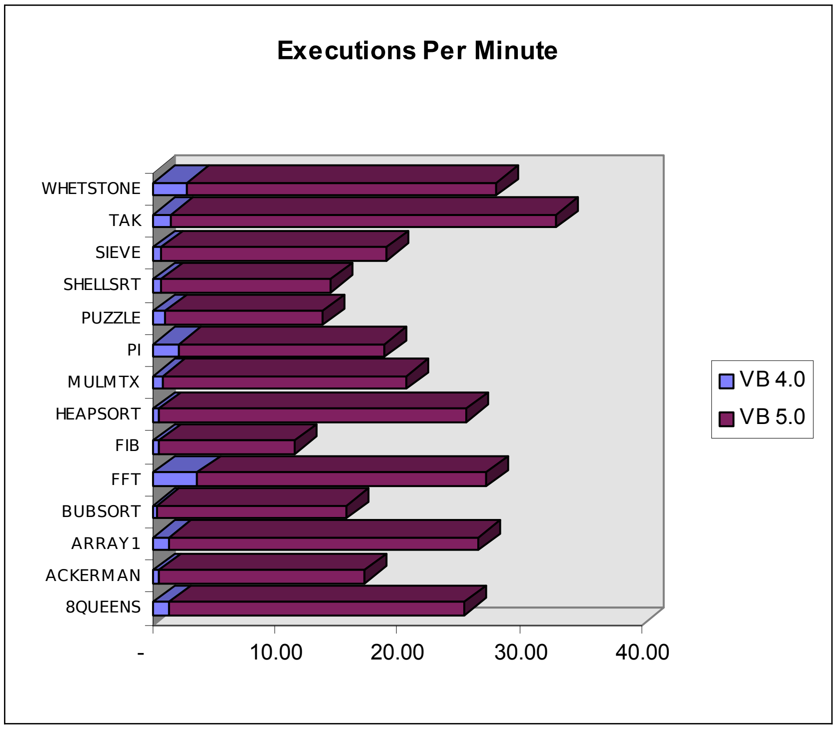
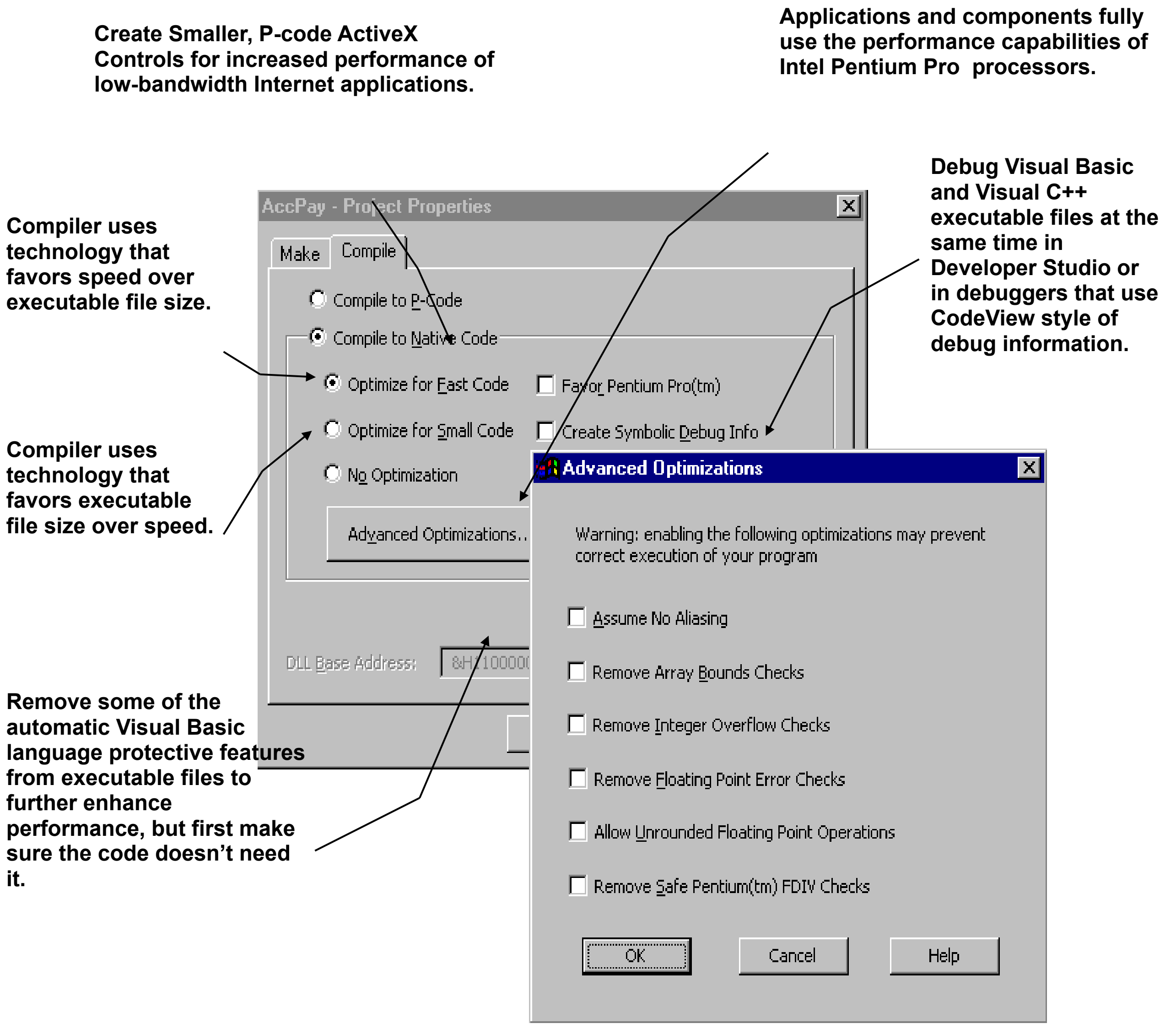


Figure 9. CPU-intense benchmark results show the advantages of Visual Basic version 5.0 over Visual Basic version 4.0.

Compiler optimization

Using technologies from the award-winning Visual C++ optimizing compiler, the Visual Basic native compiler offers capabilities not seen before in a first-generation native code compiler. This compiler should not be confused with attempts by some development tool vendors who convert syntax to that of a native compiler, and then compile code. That approach results in inefficient code that executes slowly. The Visual Basic compiler is also flexible, so that a developer can choose between optimizing code for smaller size or faster performance. Figure 10 shows compiler optimizations available to a developer.


Create Smaller, P-code ActiveX Controls for increased performance of low-bandwidth Internet applications.

Applications and components fully use the performance capabilities of Intel Pentium Pro processors.

Compiler uses technology that favors speed over executable file size.

Debug Visual Basic and Visual C++ executable files at the same time in Developer Studio or in debuggers that use CodeView style of debug information.

Compiler uses technology that favors executable file size over speed.

Remove some of the automatic Visual Basic language protective features from executable files to further enhance performance, but first make sure the code doesn't need it.

AccPay - Project Properties

Make | Compile

Compile to P-Code
[✓ Compile to Native Code]
[✓ Optimize for Fast Code] / Favor Pentium Pro(tm)
Optimize for Small Code / Create Symbolic Debug Info
No Optimization

Advanced Optimizations...

DLL Base Address: &H1100000

Advanced Optimizations

Warning: enabling the following optimizations may prevent correct execution of your program

- Assume No Aliasing
- Remove Array Bounds Checks
- Remove Integer Overflow Checks
- Remove Floating Point Error Checks
- Allow Unrounded Floating Point Operations
- Remove Safe Pentium(tm) FDIV Checks

OK | Cancel | Help

Figure 10. Visual Basic compiler optimization options.

Multi-threading made easy

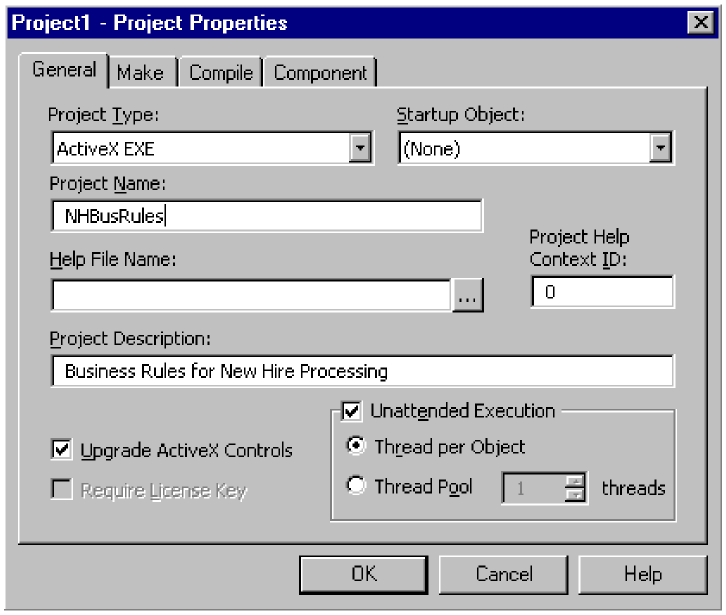
Multi-threading is one of those complex, underlying technologies that most programmers simply want to take advantage of and not worry about. ActiveX Components in distributed applications need to be multi-threaded in order to be responsive. In the past, this meant powerful servers that used an entire process for each object. Using the Unattended Execution option, a developer can create multi-threaded servers with objects on separate threads for maximum responsiveness. In addition, because of being handled by Visual Basic, a developer doesn't deal with scheduling, dead locks, and contention. Multi-threaded servers possess several characteristics that are particularly important for scalability in three-tier distributed applications and Active Server Pages for several reasons:

- Servers marked for unattended execution improve system utilization by automatically allocating object instances across multiple threads for better performance and scalability. Because they support the COM apartment-threading model, these components require fewer resources when invoked concurrently from different threads.
- Servers marked for unattended execution are thread safe, meaning objects in one apartment (thread) don't conflict with objects in other apartments, and that their methods are protected against re-entrancy.
- With servers marked for unattended execution, a developer can limit the number of threads an ActiveX Component acquires for better control over system resource.
- Servers marked for unattended execution suppress all forms of user interaction, and logs errors to the system log file instead—thus enabling a lights out operation as desired.
- Although servers marked for unattended execution were designed to improve the scalability of distributed components, it can be used for any out-of-process component.

Unattended execution creates scalable components that use the multi-threading capability of Windows NT.

Each class is automatically instantiated in its own thread.

The preset number of threads are shared by each instantiated class


Project1 - Project Properties

General | Make | Compile | Component

Project Type: ActiveX EXE

Startup Object: (None)

Project Name: NHBusRules

Help File Name:

Project Help Context ID: 0

Project Description: Business Rules for New Hire Processing

[✓] Upgrade ActiveX Controls

[ ] Require License Key

[✓] Unattended Execution

(●) Thread per Object

( ) Thread Pool 1 threads

OK | Cancel | Help

Figure 11. With Visual Basic version 5.0, developers can create multi-threaded components with unattended execution.

Language Improvements

Many language improvements in Visual Basic version 5.0 deal with object-oriented programming. The fundamental concept of object-oriented programming is *abstraction.* Earlier programming models forced a programmer to write code making specific references to particular data structures and variables. With this approach, changes to the internal data structures of a program could force changes to many of lines of code. In contrast, with object-oriented programming, a developer creates abstract representations or classes for the program parts. Only the code inside a class needs to know about the internal data structures. Other parts of the program deal with the abstract definition of the class, which is less likely to change. A primary benefit of object-oriented programming is that code maintenance is much easier because changes affect only specific classes rather than the entire program.

Other essential elements of object-oriented programming are:

- **Encapsulation**—a way of ensuring that only certain well-defined parts of a class are accessible to code outside the class.
- **Polymorphism**—the ability to implement the same abstraction in different objects. For example, two different objects could provide customer data in the same way.

Code Reuse

Code reuse is the ability to pass characteristics of one component on to another component, usually accomplished by some form of inheritance, containment, or delegation. Visual Basic version 5.0 supports abstraction, encapsulation, polymorphism, and code reuse with new language features, such as:

- **ActiveX Controls.** Controls have always been an important way to reuse code in Visual Basic, but it was always somebody else's code. Now controls can be authored in Visual Basic. A developer authoring a control in Visual Basic can use existing controls as building blocks, through containment and delegation.
- **Implements statement.** The Implements statement enables polymorphism by allowing multiple classes to respond to the same interface. For example, a program that recognizes the ICustomerData interface can communicate with objects of any class that implements ICustomerData. Classes created in Visual Basic version 5.0 can implement any number of interfaces. This powerful feature, also found in COM and the Java language, allows large systems to evolve over time instead of requiring complete redesign.

- **Interface inheritance and delegation.** When developers use the Implements statement to inherit an interface, they can also reuse the code behind the interface by delegating to the class that provided it. Properties and methods can be overridden completely, or additional code can be executed before or after delegation.
- **Property procedure improvements.** Property procedures are a powerful tool for encapsulation, allowing objects to validate their own data when property values are assigned. Now Property procedures can use all the features of Sub and Function procedures, including the new typed optional arguments.
- **Typed optional arguments.** Both private and public procedure declarations can now contain optional arguments with explicit data types, such as strings or integers, and default values that will be used if the argument is omitted. This eliminates the need to write code for data type checking or set explicit default values.
- **Enumeration Variables.** Developing applications that use custom procedures and objects is easier with enumeration variables, which exposes sets of named constants. The Visual Basic IntelliSense features can use the new data type, Enums, to show drop-down lists of constants valid for a particular argument or variable—an improvement over looking up Const declarations.
- **GlobalMultiUse objects.** Code reuse can be easier than reusing objects. Now, useful general-purpose procedures can be placed in a GlobalMultiUse class and compiled into an ActiveX DLL. Developers, and users of desktop systems such as Microsoft Office, can use the procedures in a GlobalMultiUse class without declaring an instance of the class, just as if the procedures were part of the Visual Basic language. Public Enumerations in GlobalMultiUse classes can provide global constants for these procedures.

In addition to features that enhance code reuse, Visual Basic version 5.0 includes language enhancements in other areas of programming.

Events

Any class created with Visual Basic version 5.0 can raise its own events. Events are also of major importance when authoring ActiveX Controls.

AddressOf Operator

Visual Basic now provides a way to pass the address of a call-back procedure to a Windows API, enabling the use of system timers, APIs that enumerate system objects, and window sub-classing.

Friend Methods

A reusable component authored in Visual Basic can use Friend properties and methods to communicate privately between the objects the component provides, without exposing that communication to client programs that use the component.

Internet Features

The Internet and its associated technologies offer new possibilities for the Internet as well as client/server applications. Through the Internet, people have access to information stored on computers literally around the world. They can communicate through e-mail and voice; they can make purchases using their computers.

Even so, the real power of the Internet has yet to be explored—primarily because of legacy systems. Millions of programmer years and millions of dollars in developer training are invested in existing systems. Experience tells us that wholesale replacement of existing systems simply doesn’t make sense—and wholesale retraining of employees to use this new technology would be impossible. This does not mean, however, that some benefit from Internet technology cannot be achieved today.

During this decade, developers have created software components abstracted from particular COM implementations. The benefits of COM are numerous, but perhaps most significant are flexibility and choice for use. ActiveX Components based on COM can be used in a variety of implementation technologies. For example:

- Application components developed using earlier versions of Visual Basic can easily be enhanced for use over the Internet. This is an advantage in many cases because Internet browser technology offers a single, easy-to-use interface for all applications. It can also eliminate much of the burden of installing software because of its ability to automatically install software components.

- The same ActiveX Components that can be used in Internet applications can also be used in client/server applications. This is important because in some cases more mature client/server technologies are better suited to business needs.
- Developers can provide access to legacy software using the Internet by building ActiveX Controls that can reside on an Internet server and can communicate with legacy systems.

Through ActiveX technologies, Visual Basic version 5.0 combines the possibilities of Internet technology, the existing investments in applications and resources, and the proven capabilities of client/server architectures, offering developers the ability to easily move components among different implementations.

In the next section, we discuss three ActiveX capabilities in version 5.0: the ability to create ActiveX Controls, Active Documents, and ActiveX Components.

Create ActiveX Controls

An ActiveX Control typically works as a component that plugs into an application such as Microsoft Excel, a Web page, or a programming environment such as Visual Basic, Visual C++, and so forth. ActiveX Controls also often provide a user interface and expose properties, events, and methods to the application hosting the control. Visual Basic version 5.0 makes it easy to create full-featured COM-based controls. Control features that were previously the exclusive domain of the seasoned developer using C++ are easily authored using Visual Basic version 5.0. Some of these easily implemented features include:

- Self-registering controls that automatically create the proper registry entries on client systems.
- Property pages for setting control properties like the property pages in Microsoft Office 97.
- Creating ActiveX Controls from scratch.
- Creating ActiveX Controls from other controls using sub-classing.
- Creating composite ActiveX Controls built from other controls.
- Creating data-aware controls, also known as databound controls.

The ActiveX Controls created using Visual Basic version 5.0 can exhibit these features and many more. For instance, a developer can create a simple control with a user interface in Visual Basic version 5.0 in five minutes or less. In that time, a developer can design, test, and use the control. Of course, the control is not a finished product after five minutes, but this quick turnaround does illustrate the dramatic speed with which a developer can create controls.

Controls included with Visual Basic version 5.0 are up to three times faster and one-third the size of Visual Basic version 4.0 controls because they use ActiveX technologies. In fact, there is no measurable difference between the performance of a form using a sub-classed textbox control and a form using the packaged textbox control. Application performance is limited to the speed of the slowest control, so faster controls equal faster applications. Higher-performance controls and superior form rendering mean that forms load, show, and hide up to 17 times faster than with previous versions of Visual Basic.

Capabilities of ActiveX Controls authored using Visual Basic 5.0

When constructing an ActiveX Control using Visual Basic version 5.0, a developer can draw on many new features. These new features are designed for ease of development, high performance, and functionality.

The Visual Basic version 5.0 new control-creation features extend into many areas, including interface, performance, and database interaction. Other features are provided to bring Internet functionality to ActiveX Controls in an easy-to-use manner. Still others support the development of the control itself and offer the means to shorten the development process. This list contains some of the new features that developers who build controls can take advantage of:

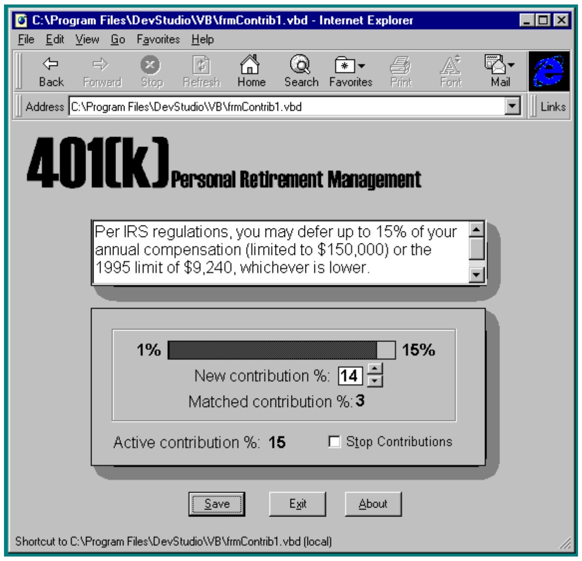
- **Data awareness in controls.** Controls can be easily linked to fields in a database.
- **Container controls.** Controls can be made to house other controls. A tabbed dialog box control or a frame control are examples of a container control.
- **Events.** Controls can fire events, dramatically simplifying programming.
- **Enumerations.** Enumerations provide support for populating the Quick Info feature of Visual Basic.

- **Default properties.** Default properties are used when a control is referred to without specifying a particular property.
- **Property pages.** Properties pages are useful for design-time configuration, or reuse of a control.
- **OCX bundles.** An OCX bundle is a single OCX file that contains multiple OCX controls.
- **Digital signatures.** Controls can easily be signed with a digital signature to ensure valid authentication.
- **Asynchronous property reading.** Asynchronous property reading allows controls, especially Internet controls, to download property settings in the background while the application remains responsive to an end user.
- **Visible/invisible at run time.** Using visible and invisible properties, a control author can decide whether the control will be visible or not at run time. The Timer control is an example of an invisible control.
- **Licensing.** Licensing offers a mechanism so programmers can protect the rights to their controls.
- **Object models.** Object models can be used to provide a sophisticated hierarchical object model within a control.
- **Transparent backgrounds.** Transparent backgrounds allow controls to host other controls without obscuring the container's graphic or text information.

Create Active Documents

What is it about intranets that corporations find so compelling? By using an intranet, information can be centralized and easily distributed. Internet technologies have made information publishing easy. Whether a document is located down the hall or in a different hemisphere, an end user can request it by typing a URL. If a user does not already have the document or if a user has an older version of the document, the latest version can be delivered automatically to the browser. Using Visual Basic version 5.0, developers can use this technology as a means to dynamically deliver entire applications and the necessary components—not just HTML-based Web pages—to a browser. Applications that are designed to automatically install and run in a browser are called *Active Documents.*

Being able to create Active Documents using Visual Basic offers numerous benefits. Rather than learning new programming languages such as Java or HTML, programmers can use their existing code bases, tools, and utilities, component libraries, and data access techniques. You can see an Active Document built using Visual Basic in Figure 12.


C:\Program Files\DevStudio\VB\frmContrib1.vbd - Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Mail
Address C:\Program Files\DevStudio\VB\frmContrib1.vbd
Links
401(k) Personal Retirement Management
1%
15%
Save
Exit
About
Shortcut to C:\Program Files\DevStudio\VB\frmContrib1.vbd (local)

Per IRS regulations, you may defer up to 15% of your annual compensation (limited to $150,000) or the 1995 limit of $9,240, whichever is lower.

New contribution %: 14
Matched contribution %: 3
Active contribution %: 15
☐ Stop Contributions

Figure 12. Active Documents are like Visual Basic forms with additional automation interfaces that allow them to be activated in a different container.

Active Documents are not an entirely new concept. You are already familiar with Microsoft Word documents. A Word document is not the same as the Word application. A Word document, with the extension .doc, contains the content, whereas the Word application, named Winword.exe, is used to create the document.

You may also know that a Word document can be viewed in other containers as long as the Word application exists on the same computer. In this case, the Word application functions like an Automation server, enabling another ActiveX container, such as Microsoft Internet Explorer, to view and activate the document. This same mechanism works for Active Documents created with the Visual Basic programming system.

Create ActiveX Components

Three-tier architecture, illustrated in Figure 13, has become a preferred programming model. Its promise of flexibility and reuse is fulfilled when components of three-tier applications are migrated to Internet applications. ActiveX Controls address component needs in the user-services tier of the three-tier architecture, but the business logic in the middle business services tier contains as many reuse possibilities.

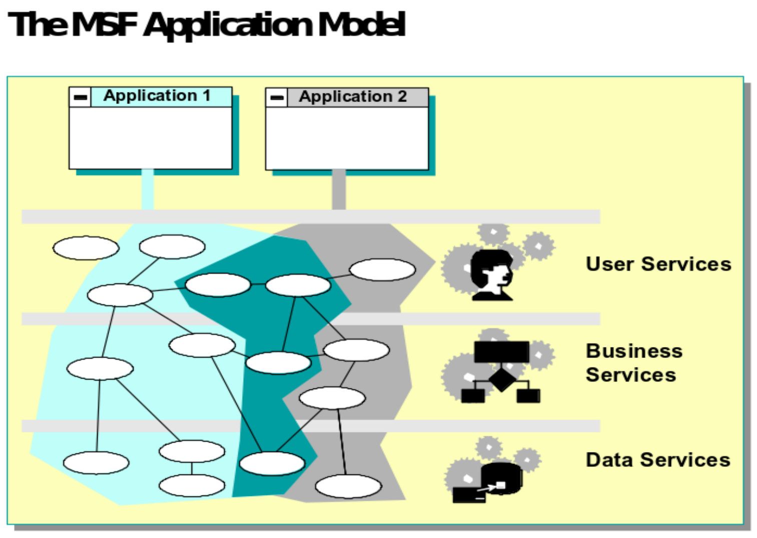


Figure 13. The three-tier architecture offers benefits in flexibility and reusability.

ActiveX technology facilitates this three-tier architecture and can be reused in traditional client/server applications, Active Server Pages, and the Microsoft Transaction Server (MTS).

ActiveX Components written in Visual Basic version 5.0 are more than multipurpose components. Here is a list of some of the new features that developers can take advantage of when building these components:

- **Versioning.** ActiveX Components can be set to project versioning, binary versioning, or no versioning. Using versioning, a developer can control how dependent components respond when changes are made to it.
- **Multi-threading.** Components can be set to run in a thread pool or to open a new thread for every object.
- **In-process and out-of-process support.** Developers can create in-process components that are significantly faster or create out-of-process components for dynamic integration of components.

- **DCOM and remote automation support.** Visual Basic version 5.0 ActiveX Server Components can be accessed from 32-bit clients using DCOM or from 16-bit clients using remote automation.
- **MTS support.** Components marked for unattended execution conform to apartment model threading to be manageable by MTS.

Extensibility, Controls, Wizards, and Builders

Today, there are hundreds of vendors building thousands of ActiveX Components and Visual Basic extensions in hundreds of categories, including multimedia, medical, factory automation, and data acquisition. By the year 2000, IDC estimates that this will grow into a billion-dollar industry. These Visual Basic extensions come in two forms.

- A developer can choose from thousands of custom controls to enhance the capabilities of applications and, with version 5.0, create their own.
- Wizards and builders, which use the Visual Basic open-object model, can be built to automate programming tasks, provide a visual interface, or simplify a complex concept.

Visual Basic version 5.0 makes advances in each of these areas by increasing the capabilities available to component vendors and to developers using Visual Basic. Virtually everything which can be built using Visual Basic version 5.0 is now exposed in the Visual Basic object model.

Using the comprehensive Visual Basic object model, a developer can seamlessly integrate extremely sophisticated technologies like object modeling. Previously, successful creation of custom controls and add-ins was limited mostly to experienced programmers using the C++ language. Now, creating controls is no longer a limitation because developers using Visual Basic can use it to create true ActiveX Controls. Creating add-ins is also no longer a limitation because the enhanced object model exposes everything a developer needs, so system-level programming is no longer required. IntelliSense makes developing wizards easier because it works with the Visual Basic object model the same way it works with other objects. The Wizard Manager displayed in the Wizards section also helps by building the base of a custom wizard for a developer.

Wizards and Builders

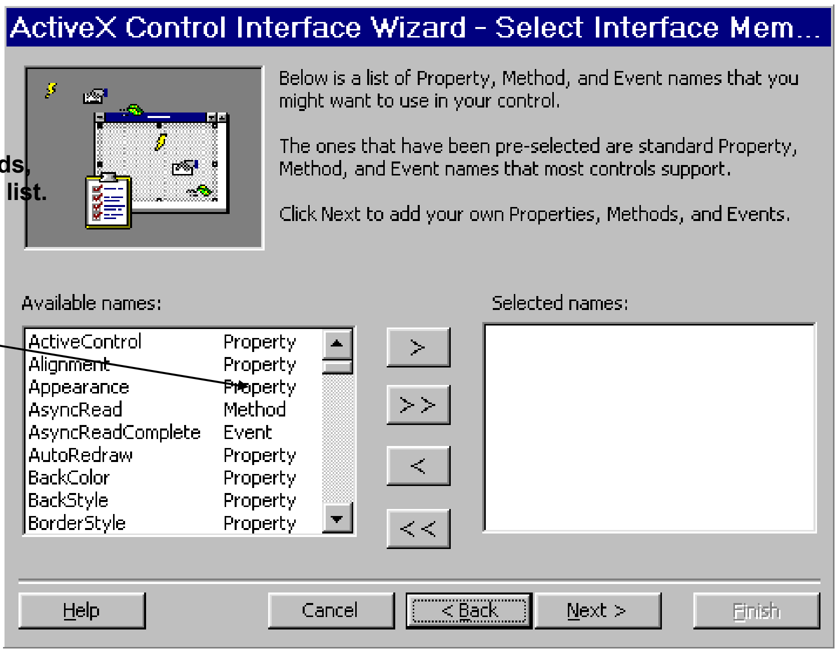
These ten wizards and builders available in Visual Basic version 5.0 can reduce complexity and increase productivity:

- **ActiveX Control Interface Wizard.** Helps developers build the public interface and functionality for a control.
- **ActiveX Document Migration Wizard.** Changes existing project forms into Active Documents.
- **Add-in Toolbar.** Developers can add wizards to a toolbar and start them by clicking a button.
- **Application Wizard.** Generates a new, fully functional application from which a developer can build a more complex application.
- **Class Builder.** Offers a visual builder that helps developers build and analyze object models.
- **Data Form Wizard.** Generates fully functional Visual Basic forms for adding, changing, and deleting records in tables.
- **Property Page Wizard.** Helps developers build property pages for controls.
- **Setup Wizard.** Contains new Internet download capabilities for ActiveX Controls and Active Documents.
- **Wizard Manager.** Helps developers build custom wizards.
- **Wizard Toolbar.** Creates a toolbar from which to launch wizards.

ActiveX Control Interface Wizard

Using the ActiveX Control Interface Wizard, illustrated in Figures 14 and 15, a developer can easily define the properties, methods, and events for a user-created ActiveX Control. For example, to select interface members for a control, a developer opens the wizard and from it adds properties, methods, and events from a list of available names.

Select standard properties, methods, and events from a list.



Below is a list of Property, Method, and Event names that you might want to use in your control.

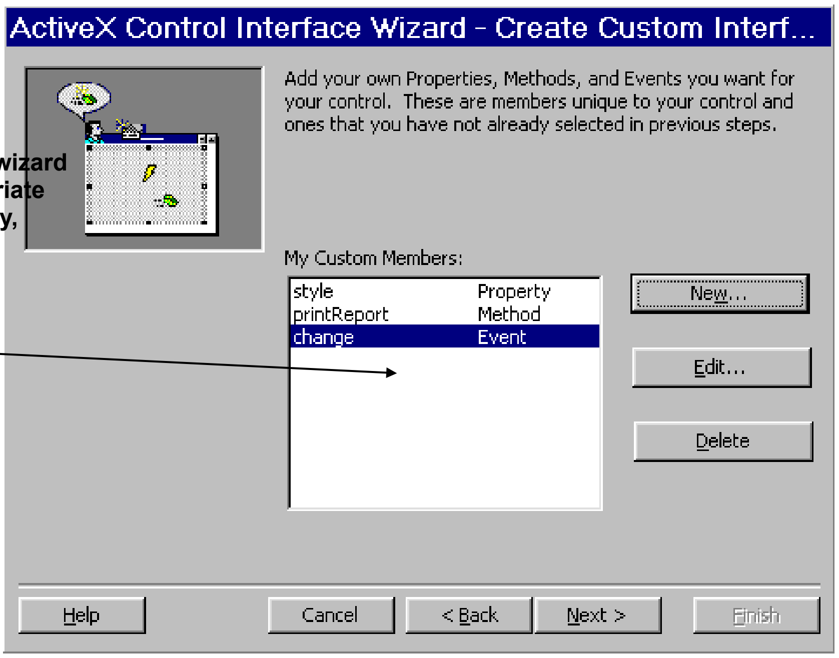
The ones that have been pre-selected are standard Property, Method, and Event names that most controls support.

Click Next to add your own Properties, Methods, and Events.

ActiveControl	Property
Alignment	Property
Appearance	Property
AsyncRead	Method
AsyncReadComplete	Event
AutoRedraw	Property
BackColor	Property
BackStyle	Property
BorderStyle	Property

Figure 14. The ActiveX Control Interface Wizard gives developers a list of properties, methods, and events for common controls.

Create a custom interface and the wizard writes the appropriate code for a property, method, or event.



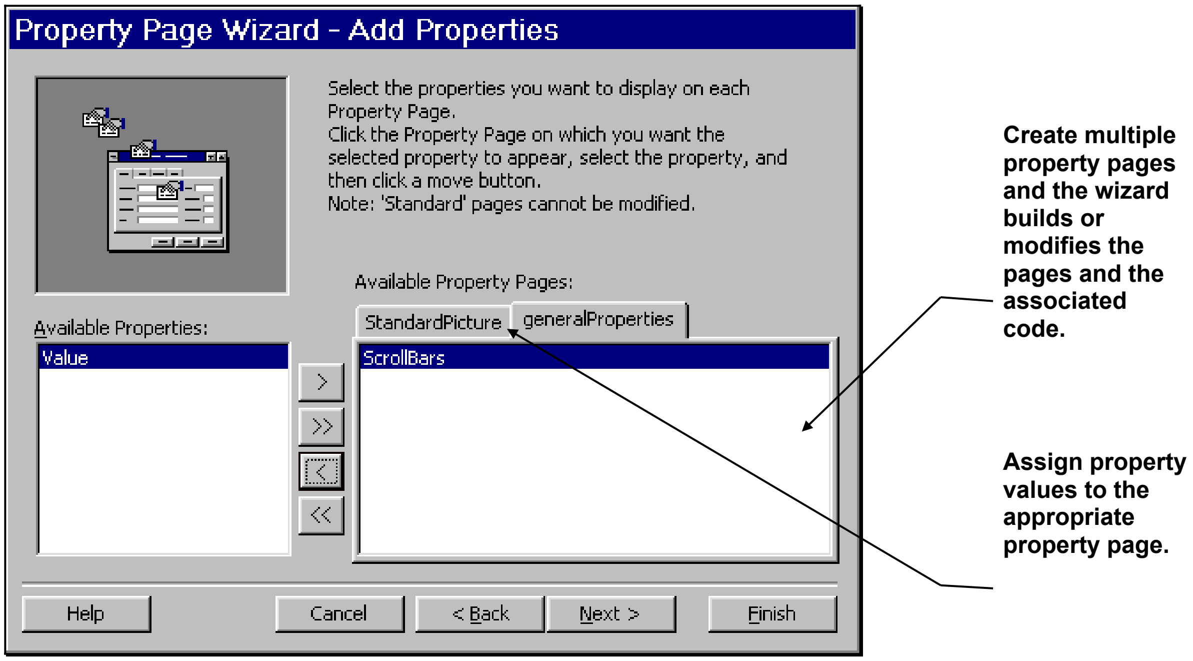
Add your own Properties, Methods, and Events you want for your control. These are members unique to your control and ones that you have not already selected in previous steps.

style	Property
printReport	Method
change	Event

Figure Figure 15. The ActiveX Control Interface Wizard makes it easy to create a custom interface.

Property Page Wizard

Using the Property Page Wizard, illustrated in Figure 16, a developer creates custom property pages automatically for a user-defined control.


Property Page Wizard - Add Properties

Select the properties you want to display on each Property Page.
Click the Property Page on which you want the selected property to appear, select the property, and then click a move button.
Note: 'Standard' pages cannot be modified.

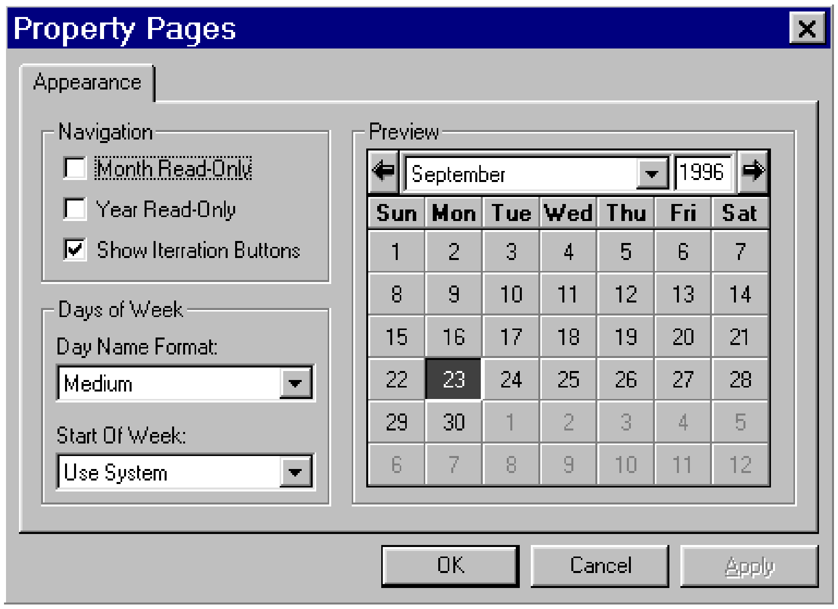
Available Property Pages:

Available Properties:

StandardPicture
generalProperties
Value
ScrollBars
>
>>
<
<<
Help
Cancel
< Back
Next >
Finish
Create multiple property pages and the wizard builds or modifies the pages and the associated code.
Assign property values to the appropriate property page.

Figure 16. The Property Page Wizard simplifies the process of creating property pages.

Figure 17 shows an example of a custom property page for a calendar control created with Visual Basic version 5.0. Regardless of the development tool hosting the control, at design time when a developer right clicks the control, this custom property page is displayed. The development tool hosting the control can be Visual Basic version 4.0, Visual Basic version 5.0, Visual C++, Microsoft Office 97, Visual FoxPro version 5.0, ActiveX Control Pad, and so forth. Custom property pages not only make ActiveX Controls easier to reuse across different development tools, they also provide a consistent, intuitive interface for the control. Note here that the custom property page can also contain a preview of how the control will look after the changes are applied.



Navigation

- [ ] Month Read-Only
- [ ] Year Read-Only
- [x] Show Iterration Buttons

Days of Week

Day Name Format: Medium

Start Of Week: Use System

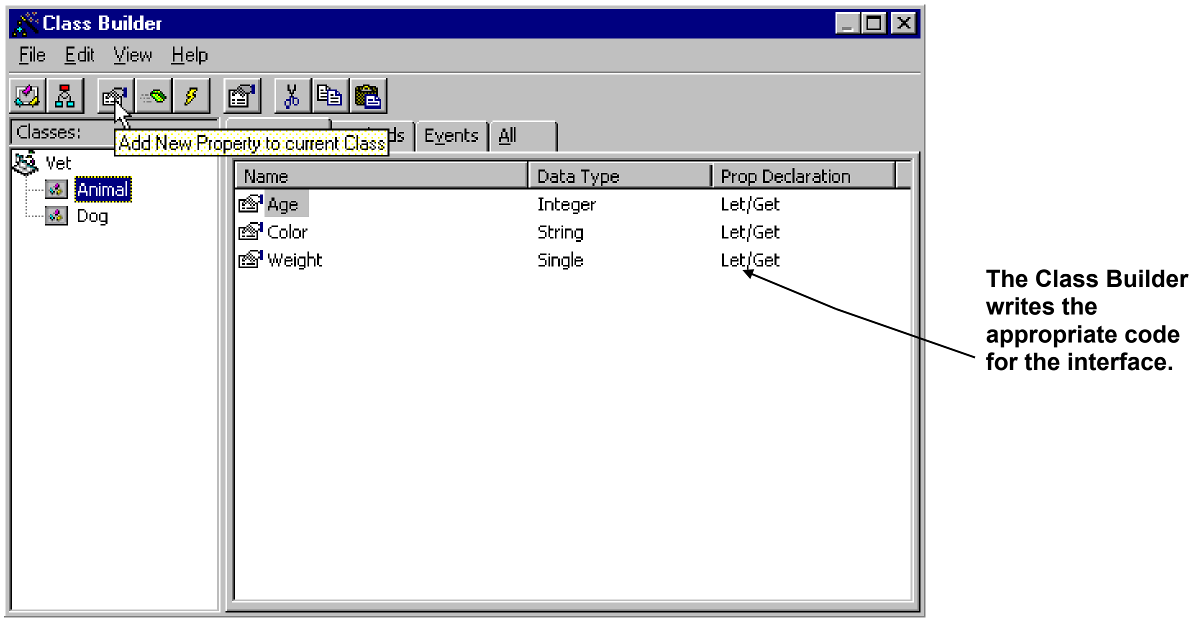
Preview: September 1996

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

Figure 17. An example of a custom property page for a calendar control.

Class Builder

Using the Class Builder, illustrated in Figure 18, a developer can view a graphical representation of the project object model and add or modify additional classes, properties, methods, and events.

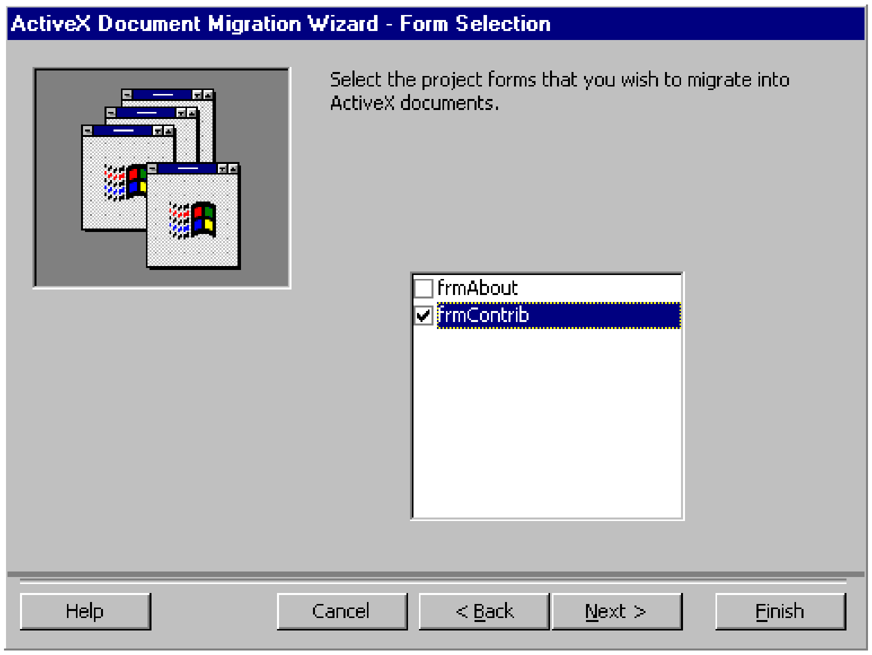

Class Builder
File
Edit
View
Help
Classes:
Add New Property to current Class
Events
All
Vet
Animal
Dog
The Class Builder writes the appropriate code for the interface.

Name	Data Type	Prop Declaration
Age	Integer	Let/Get
Color	String	Let/Get
Weight	Single	Let/Get

Figure 18.The Class Builder offers a developer a graphical representation of the project.

ActiveX Document Migration Wizard

Using the ActiveX Document Migration Wizard, visible in Figure 19, a developer can convert existing applications into Active Documents that run in the Microsoft Internet Explorer and that can be automatically downloaded and updated. Now, users can see and easily explore any application, whether it was developed for Internet, intranet, host systems, or for an existing client/server application.


ActiveX Document Migration Wizard - Form Selection

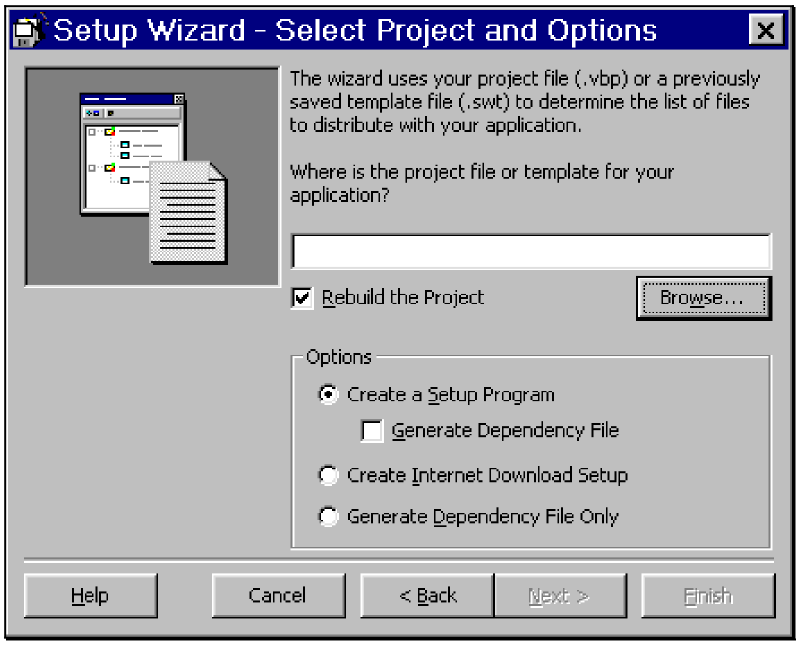
Select the project forms that you wish to migrate into ActiveX documents.

- [ ] frmAbout
- [✓] frmContrib

Figure 19. The ActiveX Document Migration Wizard.

Setup Wizard

The Setup Wizard, which you can see in Figure 20, has been enhanced to better support common installations. It supports Active Documents and ActiveX Control distribution using the Internet. It also supports installation of remote server components using DCOM and remote automation.



Setup Wizard - Select Project and Options

The wizard uses your project file (.vbp) or a previously saved template file (.swt) to determine the list of files to distribute with your application.

Where is the project file or template for your application?

[✓] Rebuild the Project

Options

[✓ Create a Setup Program]

[ ] Generate Dependency File

Create Internet Download Setup

Generate Dependency File Only

Figure 20. The Setup Wizard Select Project and Options page.

Wizard Manager

The Wizard Manager, which you can see in Figure 21, prepares a wizard template and offers a set of graphical tools a developer can use to build a custom wizard. Since the Wizard Manager is an add-in and wizards use standard project file types, building sophisticated wizards is as easy as building sophisticated applications.

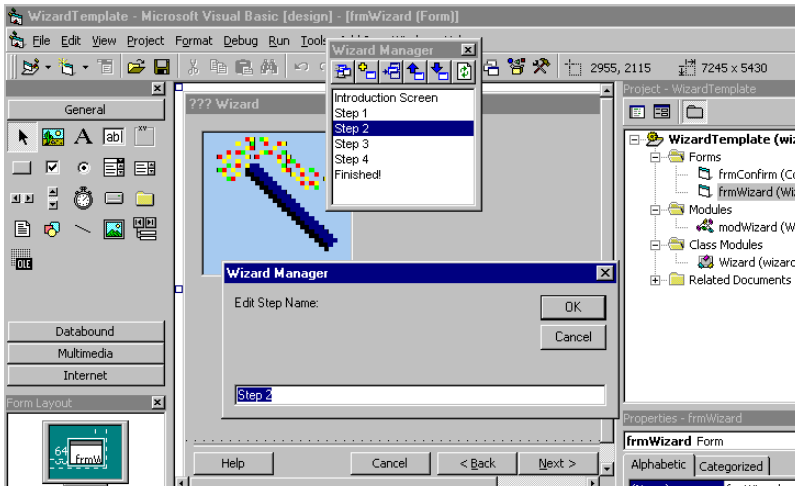


Figure 21. The Wizard Manager helps developers build custom wizards.

Enterprise Edition Features

This section of the evaluators guide focuses on Visual Basic version 5.0, Enterprise Edition, and the special set of development features it offers. Enterprise managers, architects, and developers all have their own development requirements—focused by the goal of making their customers happy and the corporation profitable. Visual Basic version 5.0, Enterprise Edition, is specifically designed to address these unique requirements by integrating virtually all of the tools and support systems needed for successful enterprise application development, deployment, debugging, tuning, and support. An enhanced feature set means that enterprise development costs are lower, well-running applications are developed more quickly, and additional staff is easier to find, use, and keep.

Essentially, the Enterprise Edition is designed to provide enhanced functionality for each phase of the client/server and Internet application life cycle:

- Database management systems (DBMS) and client/server systems planning and design including database schema and query design, and managing relationships, security, and indexes.
- Server-side procedure prototyping, benchmarking, development, and testing.
- Client application design, development, testing, and deployment.
- Three-tiered business object design, testing, and deployment. This also includes remote object management and transaction parenting.
- Transact SQL and other SQL query development and interactive debugging.

To implement these features, the Enterprise Edition integrates the following major components into Visual Basic version 5.0. You can read more about each component in other sections of this evaluators guide.

- Microsoft SQL Server version 6.5, Developer Edition.
- Microsoft Transaction Server.
- Transact SQL debugger.
- Microsoft Visual Database Tools.
- Enhanced distributed component facilities.

- Microsoft Visual SourceSafe™ version 5.0.
- Additional data access features, including Remote Data Objects 2.0, Data Access Objects version 3.5, User Connection Designer, and Application Performance Explorer.

Microsoft SQL Server Version 6.5, Developer Edition

One of the most substantial and important elements in the Visual Basic version 5.0, Enterprise Edition, is the Developer Edition of Microsoft SQL Server version 6.5. By including this world-class database management system, developers can immediately begin building databases to prototype and evaluate production systems based on a full-featured version of Microsoft SQL Server. This version, created for developers writing in Visual C++ and Visual Basic, also includes the SQL Server Programmer’s Toolkit, comprehensive Books Online documentation, and the full SQL Server tool set including SQL Enterprise Manager. This means that developers can build and test the entire database-management system without disturbing their live production database. This also eliminates the need to prototype using an ISAM or flat-file DBMS in anticipation of eventually upgrading to a full relational model running on SQL Server.

Microsoft Transaction Server

In order to implement today’s sophisticated three-tiered application designs, significant functionality must be implemented to deal with complex object and transaction management issues. And this code must be incorporated in each application that expects to access these shared, remotely managed objects. These issues include pool management, object brokering, thread coordination, and handling cross-component or cross-server two-phase commit issues. Too often, enterprise development teams attempt to take on this challenging task themselves. While it is possible to build this kind of custom system using Visual Basic, it can be a daunting task and must be supported in-house. Another serious issue is standards. If the corporation’s developers take on this task, they may bear the burden of coordinating similar efforts in other parts of the organization. It also means creating training and documentation for the approach and implementation—often from scratch.

To reduce or eliminate the need to develop these custom in-house solutions, Microsoft Transaction Server is included in the Visual Basic version 5.0, Enterprise

Edition. This sophisticated transaction and object management system removes much of the complexity from building sophisticated three-tiered server applications. It does so by providing a clear and easily coded interface to remote objects and the transactions used to manage data integrity. This means lower development and support costs, and faster time-to-market for distributed component applications as well as higher database reliability.

Microsoft Transaction Server is an essential element of the Microsoft Internet and intranet application strategy. Microsoft Transaction Server is based on proven transaction-processing methods, but its importance transcends the domain of transaction processing monitors. It defines a general-purpose programming model for distributed component-based server applications.

Even simple applications deal with transactions. However, as applications grow in sophistication, the number and complexity of transactions increases. In many cases, increases in transaction volume cannot simply be handled by a client application, but must be coordinated throughout the system. For example, when working with a customer order, a developer checks for stock on hand before debiting warehouse inventory and then checks that the customer has an account in good standing before processing a credit order.

Microsoft Transaction Server is specifically designed to support server applications as they scale over a wide range of users, from small single-user systems to high-volume Internet servers. It provides the robustness and integrity traditionally associated with high-end transaction processing systems.

The Microsoft Transaction Server application model is based on COM and offers these benefits:

- The ability to communicate with an object transparently across process and computer boundaries. This means objects can reside anywhere the network or the Internet can find them.
- The mechanism by which a component can be identified and dynamically loaded and executed. Microsoft Transaction Server handles loading and executing logical business objects for an application—so a developer does not have to write the extra code to accomplish this step.

- Microsoft Transaction Server provides an architecture by which objects can support multiple interfaces. This provides a way for clients to query an object about its support for a specific interface, thus allowing components to offer varying levels of function and to gracefully introduce new versions.
- The ability to create component-based applications by using common programming languages and tools, including Visual Basic, Java™, Visual C++, and other languages.

Microsoft Transaction Server extends COM to provide a general server application framework. In addition, Microsoft Transaction Server:

- Handles server registration, processes, and thread management; context management; the management and synchronization of shared resources; and component-based security.
- Introduces transactions into the programming model as a mechanism to achieve atomic updates and consistency across components, database systems, and network boundaries.
- Provides management tools for administering the distributed environment.

Microsoft Transaction Server can relieve many responsibilities involved with managing logical business objects that a developer uses to build n-tiered applications.

Distributed Component Object Model

Visual Basic version 5.0 and Microsoft Transaction Server use ActiveX Components based on the Component Object Model (COM) and the Distributed Component Object Model (DCOM) to communicate with each other and the operating system—regardless of where the object resides. This means that an application need not treat local in-process, local out-of-process, or remote out-of-process objects differently.

First introduced in Visual Basic version 4.0 as Network OLE, DCOM has been adopted by Visual Basic version 5.0, Enterprise Edition, as the preferred remoting transport through all Remote Automation (RA) utilities. Note that RA is supported for 16-bit applications so they can connect with 32-bit applications built using Visual Basic version 5.0. This means that applications developed using RA in Visual Basic version 4.0 can be compiled and executed using Visual Basic version 5.0. These applications can also use the same servers as applications that use 32-bit DCOM. It also means that 16-bit Windows applications can use 32-bit components created in Visual Basic version 5.0 by accessing them through a DCOM interface.

Data Access Features

A developer spends hours working through low-level data access issues deciding how to formulate intelligent questions to pose to a database server. Sometimes these questions cause a client/server application to succeed or fail because they affect scalability, maintainability, performance, and overall user acceptance of an application. Visual Basic 5.0 makes it easy to identify the most efficient query and ways to get results from it quickly. More than 80 percent of applications written using Visual Basic involve data access, and a significant number of these implement client/server designs. Because of this, RDO has played an expanding role in the Visual Basic data access strategy—especially for remote data sources.

Remote Data Objects version 2.0

RDO 2.0 provides an easy-to-learn and powerful interface to intelligent ODBC data sources by offering one of several enabling interfaces that provide easy access to complex API interfaces like ODBC. RDO 2.0 does more than provide a simplified interface to the ODBC API. It also eliminates most of the problems often encountered when coding directly to an API interface such as parameter setup, Unicode, memory management, and error-handling issues. RDO has also been integrated with a number of other data access tools and interfaces to reduce coding and simplify complex query setup and result set management tasks. Because of this, RDO 2.0 is better prepared to address the special needs of enterprise developers. New RDO features include:

- **Improved performance.** Independent benchmarks by Carnegie Technologies show 2.0’s performance improved from 38.8% to 47.6% over RDO 1.0. This means that the speed of RDO 2.0 rivals that of the ODBC API without the difficulty associated with an API interface.

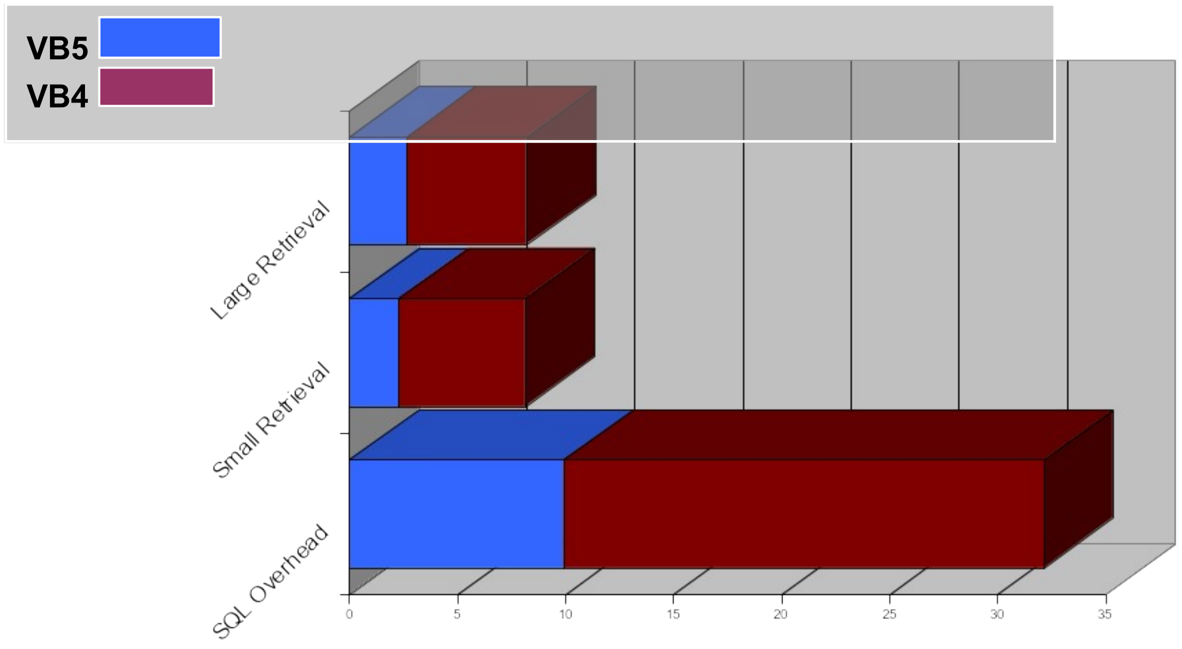


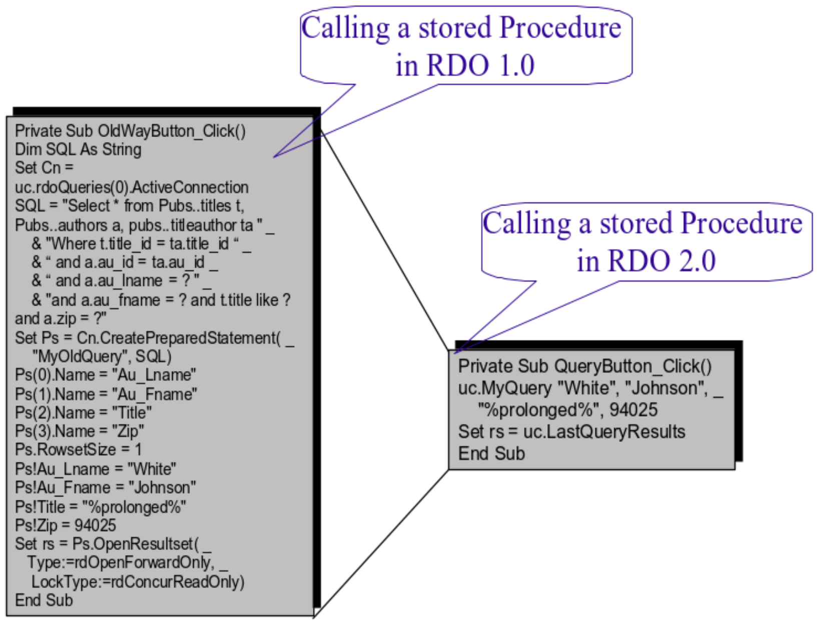
Figure 22. RDO 2.0 is up to 47 percent faster than RDO 1.0.

- **An entirely new, event-driven programming model.** RDO exposes a variety of events to support common error, connection, and query processing handlers as well as several special-case operations. This means that both asynchronous and synchronous applications need no longer block or poll while waiting for operations to complete, and can address far more sophisticated design paradigms. See the Event-Driven Programming Up Close section for details.
- **Complete asynchronous operations coverage.** RDO 2.0 expands earlier asynchronous coverage to virtually all data access operations. This means that code need no longer block while waiting for connections, cursor population, or reconstruction. This also equates to smoother-running applications that better utilize the ability of Windows to work in the background while other operations wait for external resources. Users also benefit, because the time spent watching hourglass cursors while waiting for data access operations to complete can be reduced.
- **The new Client Batch cursor library.** This new library was developed by the Visual FoxPro team to improve client-side cursor performance and to enable

optimistic batch updates. This technology permits update operations to be marshaled off-line and submitted on a batch basis. In case of collisions, the Client Batch cursor library manages errors and enables a number of sophisticated update management schemes. The result of this new approach to cursor management is higher performance and simpler programming—even for complex batch update operations.

- **Optimistic batch updates.** Developers know that batch transactions save processing time by grouping an entire set of related operations into a single batch. Batch processing virtually eliminates the overhead associated with individual operations, thus improving performance. Using this technology, applications can fetch data in a single operation and postpone committed update operations until a logical batch can be submitted. This technique greatly simplifies and standardizes collision handling, thus improving performance and scalability. The new Client Batch cursor library is specially designed to implement this technology against a variety of ODBC data sources. It also supports the concept of dissociate connections so that while the client is building the logical batch, a connection to the remote server need not be maintained.
- **The UserConnection Designer.** While the UserConnection Designer is not really a part of the RDO 2.0 interface, the designer is implemented using Visual Basic version 5.0 RDO 2.0 technology. The UserConnection objects the designer creates insulate developers from the rigors of underlying data access operations and support a higher degree of integration in team development efforts. For example, an individual developer can use the UserConnection Designer to create a universal business object interface that all developers contributing to a project can access. As the need arises, the UserConnection object can be tuned to address changing business needs without affecting other components of the application. Since the UserConnection can point to stored procedures—even complex, multi-parameter queries—updating server-based stored procedures can virtually eliminate the need to update applications when a business criterion evolves. This enhanced query management technology also eliminates the need to code and debug complex ODBC call syntax queries or to hand-code query parameters. One benefit is that repetitive query code is reduced by a factor of 10 to 1, as seen in the example in Figure

23, while developers execute complex queries as methods. These UserConnection objects can be constructed and used just as any other class library component, thus permitting applications to easily leverage company-wide standard queries.


Calling a stored Procedure in RDO 1.0
Calling a stored Procedure in RDO 2.0

```
Private Sub OldWayButton_Click()
Dim SQL As String
Set Cn =
uc.rdoQueries(0).ActiveConnection
SQL = "Select * from Pubs..titles t,
Pubs..authors a, pubs..titleauthor ta " _
   & "Where t.title_id = ta.title_id " _
   & " and a.au_id = ta.au_id _
   & " and a.au_lname = ? " _
   & "and a.au_fname = ? and t.title like ?
and a.zip = ?"
Set Ps = Cn.CreatePreparedStatement( _
   "MyOldQuery", SQL)
Ps(0).Name = "Au_Lname"
Ps(1).Name = "Au_Fname"
Ps(2).Name = "Title"
Ps(3).Name = "Zip"
Ps.RowsetSize = 1
Ps!Au_Lname = "White"
Ps!Au_Fname = "Johnson"
Ps!Title = "%prolonged%"
Ps!Zip = 94025
Set rs = Ps.OpenResultset( _
  Type:=rdOpenForwardOnly, _
   LockType:=rdConcurReadOnly)
End Sub
```

```
Private Sub QueryButton_Click()
uc.MyQuery "White", "Johnson", _
   "%prolonged%", 94025
Set rs = uc.LastQueryResults
End Sub
```

Figure 23. Sample repetitive query code reduced. This example shows how RDO 2.0 can be used to run a stored procedure by executing a method against the UserConnection object. All of the query building and configuration tasks have already been coded at design time and are imbedded in the UserConnection object—this can result in a 10 to 1 reduction in run-time code.

- **Enhanced stored procedure management.** Increasingly, n-tiered client/server designs do not expose database tables to a development team. This approach is needed to support the scalability and security requirements dictated by the architecture. To access the underlying data, DBMS systems administrators provide gated access to custom stored procedures that perform the business logic and data manipulation chores. Because of this, complete support for stored procedures is a requirement of any serious client/server front end—especially in larger systems.

  To meet this requirement, RDO 2.0 strengthens its already robust stored procedure handers with new features for SQL Server and other ODBC data sources. For example, working with the UserConnection Designer, stored procedures can be exposed as application defined connection methods. Now, developers using RDO 2.0 can objectize stored procedures so they are exposed as methods against the UserConnection object—parameters and all. Developers no longer need to code trouble prone ODBC call syntax queries or even construct custom RDO query objects in code at run time. The result is faster, cleaner code that works correctly the first time and code that can be distributed company wide as a standard class library.

- **Event-driven programming up close.** For the first time, RDO 2.0 introduces event-driven data access programming to developers who need more control over synchronous and asynchronous operations. This means that applications can easily use other execution threads in a system, including those in the application being built, those in Windows, and their own application's windows, those in ODBC drivers, or threads running on remote engines such as SQL Server—without complex thread programming. For example, developers can start opening an remote data object as the application is loading and not worry about being blocked or having to poll for completion of the operation. When using this RDO 2.0 feature, forms load and display faster, and users experience smoother running applications that are more quickly ready to accept input.

  Another significant aspect of this innovation is the ability to intercept operations before they are executed and after the operation has completed. For example, applications can intercept cursor update operations and substitute stored procedure calls in cases where direct table access is not permitted. This makes operation delegation far simpler—or possible—an essential n-tier design requirement. Event-driven programming means a new level of control over:

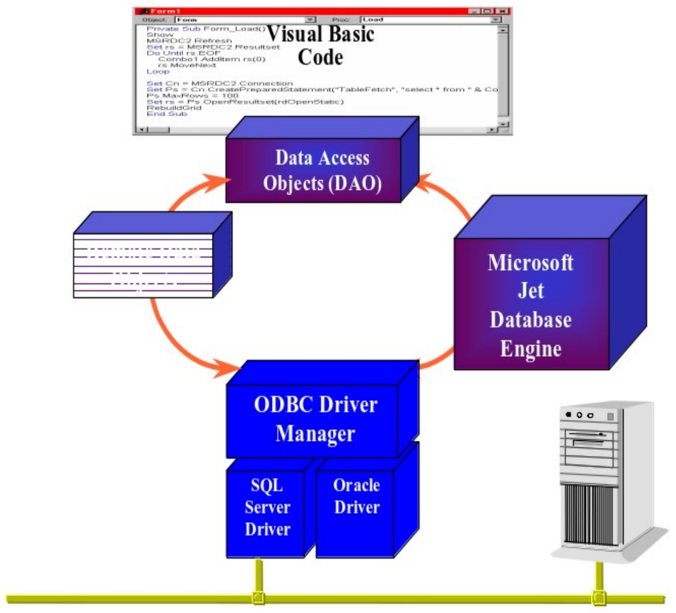
  - **Connections.** Before a connection is attempted, the WillConnect event fires to indicate that a connection is about to be attempted. Here, a developer can modify the connection parameters, perform support operations, or simply deny the operation. Once a connection operation is complete, synchronously or asynchronously, the Connect event fires to indicate the success or failure of the operation. When the connection is closed, the Disconnect event is fired.

- **Result sets.** When it comes time to create a result set, WillExecute fires before the query is executed so a developer can substitute a replacement operation or simply log the operation. Among other events used to monitor and control all aspects of query execution is the QueryTimeout event so a developer can include logic to keep waiting for the query to complete. This is useful for times when the network performance is slow.

- **Error and message handling.** SQL Server often returns informational messages to an application. While these are important, they can be difficult to separate from the more serious error message traffic returned over the same interface. To simplify the quandary, a developer can use the InfoMessage event to indicate when an informative, non-error message arrives. Other errors are returned in the RDO error collection that already fires the general trappable error event handler in Visual Basic. This is easier to handle than the technique required by the ODBC API.

Data Access Objects version 3.5

Data Access Objects (DAO) has provided the foundation for all native Microsoft Jet database and ISAM support since Visual Basic version 3.0. Over time, a number of enhancements have increased the speed and intelligence of DAO when dealing with all types of data—including ODBC data sources. For Visual Basic version 5.0, DAO version 3.5 adds a significant new feature that will fundamentally change the way DAO is used and perceived—ODBCDirect. This new mode enables an application to open a Workspace object that is either connected to the Jet database engine by default or to RDO 2.0. In either case, a developer can continue to use Data Access Objects.



```
Private Sub Form_Load()
Show
MSRDC2.Refresh
Set rs = MSRDC2.Resultset
Do Until rs.EOF
   Combo1.AddItem rs(0)
   rs.MoveNext
Loop

Set Cn = MSRDC2.Connection
Set Ps = Cn.CreatePreparedStatement("TableFetch", "select * from " & Co
Ps.MaxRows = 100
Set rs = Ps.OpenResultset(rdOpenStatic)
RebuildGrid
End Sub
```

Figure 24. With DAO version 3.5, the DAO object model is no longer bound exclusively to the Jet database engine.

- **ODBCDirect.** ODBCDirect delegates calls to the equivalent RDO functionality so developers can write to both Jet and to RDO using the same family of objects, collections, properties, and methods. DAO is a widely used and familiar database programming model, so ODBCDirect gives developers the best of both worlds: the familiarity of the DAO object model combined with the performance of RDO. This way, developers can leverage existing DAO code to tap into the power and flexibility of RDO when accessing remote data sources.

- **DAO/Jet connectivity.** Connecting to an ODBC data source through a Microsoft Jet database is still a useful and important connection model. All of the DAO version 3.0 features for Jet are still supported in DAO version 3.5. By going through Jet, a developer has access to other desktop database formats such as Paradox, dBase, Microsoft Visual FoxPro, Microsoft Excel and Lotus spreadsheets, and text files, as well as earlier versions of Jet .mdb files. Further, a developer can join tables from two or more different ODBC sources and from different desktop database formats in complex queries, with a fully updateable result set.

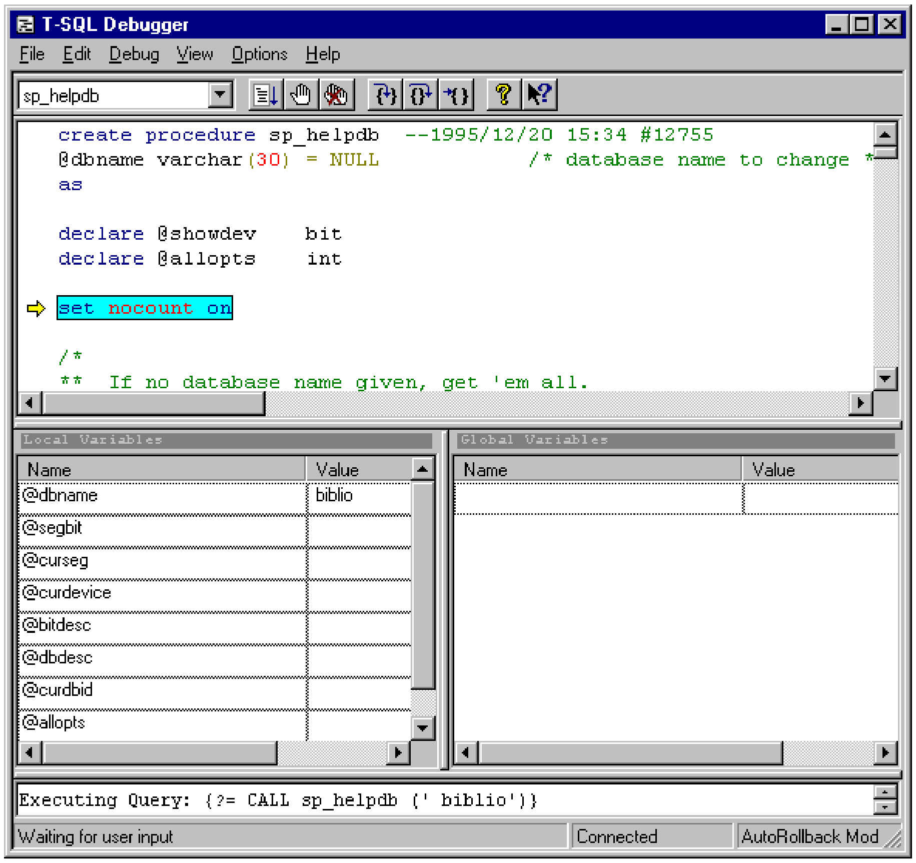
Transact SQL Debugger

Data access architects encourage developers to implement business logic and referential integrity constraints on a server using stored procedures. While this approach is easy to discuss, it is not always easy to debug—especially when it comes time to find out what is going on with a procedure. So historically, server executed code ran with virtually no feedback mechanisms, and unless additional feedback mechanisms were inserted into procedures, developers could not easily determine if or when stored procedures ran and whether they succeeded or failed.

SQL Server version 6.5 changed all of this by exposing an interface that can be accessed through Remote Automation and Visual Basic. Now, when a client runs a stored procedure—by whatever means—SQL Server version 6.5 passes back information that tracks the stored procedure code and execution progress. This new interface also supports single-stepping the stored procedure and inserting breakpoints. This new functionality gives interactive control over server-side procedure execution.

Visual Basic version 5.0 capitalizes on this technology with its own integrated Transact SQL Debugger (T-SQL debugger). For the first time, developers can step through stored procedure code just as if it was application resident Visual Basic code. This makes end-to-end debugging possible. Developers can see the server-side procedure source, set and trap breakpoints, examine and modify server-side local variables, and examine server-side global variables. As the stored procedure logic branches to other called stored procedures, the T-SQL Debugger follows along—visually tracing the execution.

This means that when code executes an UPDATE statement through an updateable cursor or with an Execute method, Visual Basic can automatically launch the T-SQL Debugger, which you can see in Figure 25. Using the T-SQL Debugger, a developer can step through the triggers being executed on the server. If the code uses business objects implemented with stored procedures, developers can also step through those as they are invoked from Visual Basic code or interactively.



```
create procedure sp_helpdb  --1995/12/20 15:34 #12755
@dbname varchar(30) = NULL              /* database name to change *
as

declare @showdev    bit
declare @allopts    int

set nocount on

/*
**  If no database name given, get 'em all.
```

Name	Value
@dbname	biblio
@segbit	
@curseg	
@curdevice	
@bitdesc	
@dbdesc	
@curdbid	
@allopts	

```
Executing Query: {?= CALL sp_helpdb (' biblio')}
```

Figure 25. Developers can use the Transact SQL Debugger to step through stored procedures or triggers executed on SQL Server.

Microsoft Visual Database Tools

Microsoft Visual Database Tools (VDT) offer the ideal visual RAD environment for working with databases in a client/server environment. Using Data View, developers can connect to multiple databases and work with database objects. The Database Designer for SQL Server version 6.5 databases directly attaches to the database so developers can visually design tables and relationships. Using the Query Designer, a developer can quickly create SQL queries against any ODBC data source, and view and edit the test data.

Microsoft Visual Database Tools work in concert with existing database tools. For example, developers still use the SQL Server Enterprise Manager to set up security on objects, perform backup tasks, transfer databases, and perform other administrative tasks.

Data View

The Microsoft Visual Database Tools Data View feature offers a visual interface to the collection of databases being accessed within a client/server project. Data View establishes a live connection to each database so developers can work directly with the various database objects during development. For example, a developer can open any database to view tables, defined views, and stored procedures. Data View provides detailed information about objects and properties within each database, including table definitions and field types, key structures, and stored procedure code. Data View coordinates with the Query Designer and Database Designer features, and provides a tightly integrated database development and maintenance system. Data View works against any ODBC-compliant database and can show multiple connections against heterogeneous databases.

In summary, Microsoft Database Tools Data View is implemented to:

- Work with multiple database connections.
- Open and browse tables in a database.
- Create and edit stored procedures and triggers for SQL Server and Oracle databases.
- Edit table definitions for SQL Server version 6.5 database tables.
- Create database diagrams for SQL Server version 6.5 databases.
- Create queries against any ODBC data source.

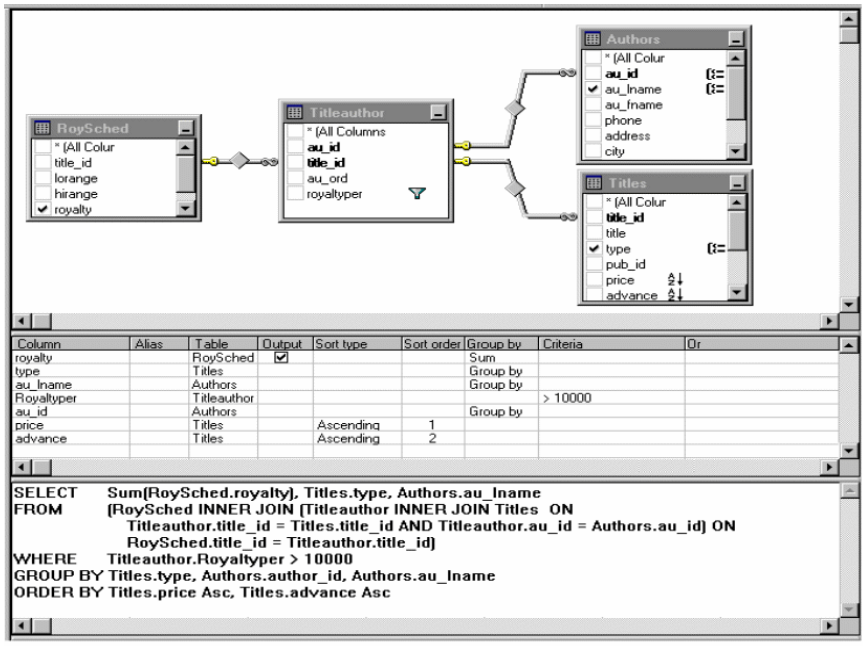
The Query Designer

Microsoft Visual Database Tools also provide a sophisticated SQL Query Designer that works against any ODBC data source. This GUI interface takes the development complexity out of even the most sophisticated SQL queries. The integrated Query Designer:

- Provides an easy-to-use interface for visually constructing the simplest to the most complex SQL statements.
- Generates the data-manipulation language (DML) for Select, Insert, Update, and Delete queries for use in program code or stored procedures.

- Exposes live views of data sources so a developer can drag tables directly into the Design pane of the Query Designer window to build queries. As a developer selects columns from the selected tables, the SQL pane reflects the dynamically constructed SQL statement.
- Supports direct modification of the SQL statement; changes are reflected in the Design pane. This supports development or importation of sophisticated SQL statements that might be more difficult to create using the GUI interface.
- Test executes any SQL statement and views the results in a Results pane.
- Easily creates complex queries across multiple tables and databases, automatically creating SQL joins and visually depicting these relationships in the Design pane.

The SQL pane also is a live pane and can be used to create stored procedures, to execute arbitrary database definition language (DDL) commands directly against any ODBC data source, or to perform ad hoc SQL queries. Microsoft Visual Database Tools thus provide a complete, tightly integrated database development environment for developers building client/server applications.



Column	Alias	Table	Output	Sort type	Sort order	Group by	Criteria	Or
royalty		RoySched	✓			Sum		
type		Titles				Group by		
au_lname		Authors				Group by		
Royaltyper		Titleauthor					> 10000	
au_id		Authors				Group by		
price		Titles		Ascending	1			
advance		Titles		Ascending	2			

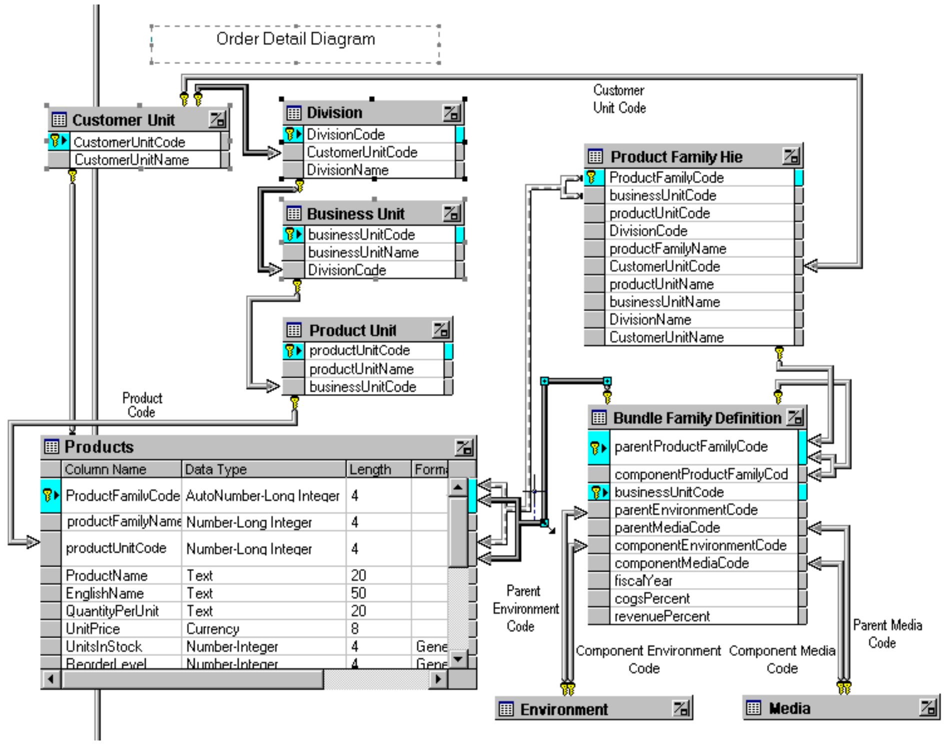
```
SELECT     Sum(RoySched.royalty), Titles.type, Authors.au_lname
FROM       (RoySched INNER JOIN (Titleauthor INNER JOIN Titles  ON
              Titleauthor.title_id = Titles.title_id AND Titleauthor.au_id = Authors.au_id) ON
              RoySched.title_id = Titleauthor.title_id)
WHERE      Titleauthor.Royaltyper > 10000
GROUP BY Titles.type, Authors.author_id, Authors.au_lname
ORDER BY Titles.price Asc, Titles.advance Asc
```

Figure 26. Developers can use the Query Designer to graphically create query definitions against any ODBC data source.

The Database Designer

Microsoft Visual Database Tools also provide a complete Database Designer for users of Microsoft SQL Server version 6.5. The Database Designer is based on an extensible architecture so that support for other database systems can be added in the future. Using the Database Designer, database administrators and developers can visually create new SQL Server databases and can modify the structure and properties of existing SQL Server databases without writing a single SQL statement. The Database Designer offers ease-of-use features, such as those offered in Microsoft Access, to SQL Server databases. Comparatively, what used to take developers hours to accomplish can now be completed with just a few mouse clicks. For example, a developer can use the Database Designer to change the data type on an SQL field, from type **Char** to a type **Int,** with a choice from a shortcut menu. With this selection, the field type is changed, which previously would have required manual DDL operations to export the entire table, drop the table, create a new table with the new data type, and import the data into the new table. To simplify and standardize this process, the Database Designer can generate scripts of DDL that are submitted to database administrators for review and execution in controlled database environments.

The Database Designer can also be used to set up database designs with primary and unique keys, check constraints, column properties, indexes, and foreign key relationships between tables. Using the Data View, developers can create triggers, cascading updates, inserts, and deletes. Because developers can functionally group tables, database diagrams can be dragged into the Query Designer to quickly construct queries against these logical groupings.



Column Name	Data Type	Length	Form
ProductFamilyCode	AutoNumber-Long Integer	4	
productFamilyName	Number-Long Integer	4	
productUnitCode	Number-Long Integer	4	
ProductName	Text	20	
EnglishName	Text	50	
QuantityPerUnit	Text	20	
UnitPrice	Currency	8	
UnitsInStock	Number-Integer	4	Gene
ReorderLevel	Number-Integer	4	Gene

Figure 27. The Database Designer simplifies the process of creating and managing SQL Server databases.

Visual SourceSafe Version 5.0

As an additional aid in team development projects, Visual Basic version 5.0, Enterprise Edition, includes Microsoft Visual SourceSafe version control software version 5.0, an enhanced version control and configuration management system. Visual SourceSafe version 5.0 includes features to facilitate Web site content management and enhanced version control features to aid in managing software development.

Visual SourceSafe version 5.0 Features

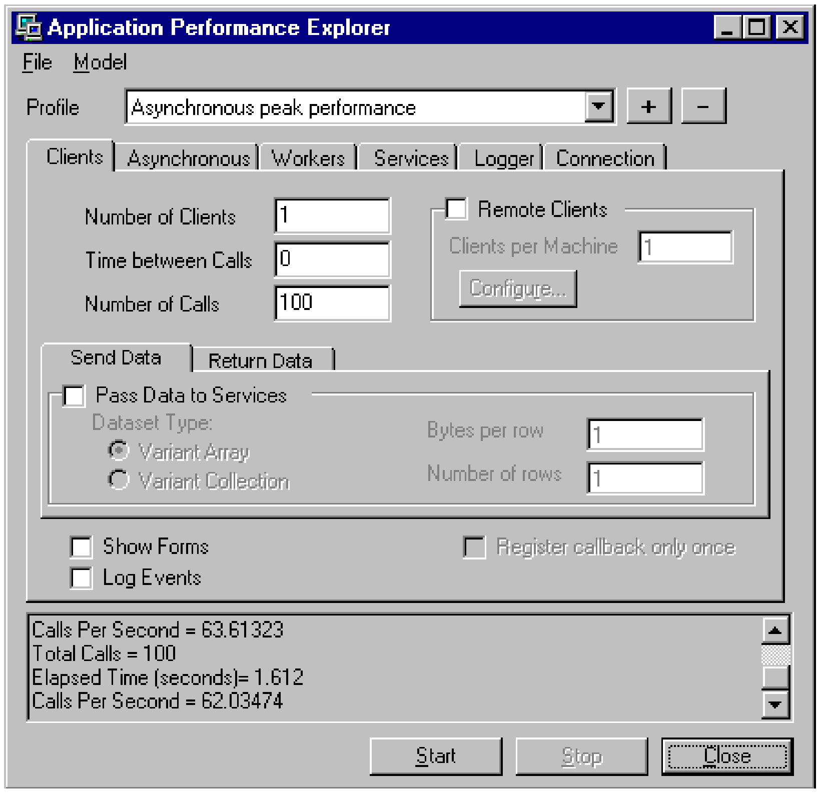
- Visual Merge for point-and-click conflict resolution.
- An enhanced Project Difference, which keeps team members in sync by comparing project versions.
- An Archive utility, which can be used to move projects between separate Visual SourceSafe databases or to archive unnecessary projects.

Visual SourceSafe Project-oriented Version Control

Visual SourceSafe works at both the file and project level, offering a unique project oriented approach to version control that promotes component reuse while reducing the complexity of the development environment. Visual SourceSafe versions files and projects efficiently while preventing accidental loss or alteration of content. These features make it an ideal product for professional software and Web site development teams that require the tools necessary to manage change in a team environment.

The Application Performance Explorer

The Application Performance Explorer (APE), illustrated in Figure 28, is a utility to aid in the design, deployment planning, and performance tuning of distributed client/server applications. By helping a developer perform component architecture performance simulation, APE makes it easy to run automated what-if tests to profile the performance of a multi-tier application in different network topologies.



Profile: Asynchronous peak performance

Number of Clients: 1
Time between Calls: 0
Number of Calls: 100

Remote Clients
Clients per Machine: 1

Pass Data to Services
Dataset Type:
Variant Array
Variant Collection
Bytes per row: 1
Number of rows: 1

Show Forms
Log Events
Register callback only once

```
Calls Per Second = 63.61323
Total Calls = 100
Elapsed Time (seconds)= 1.612
Calls Per Second = 62.03474
```

Figure 28. The Application Performance Explorer makes it easier to design distributed client/server applications, plan their deployment, and fine-tune their performance.

In addition, using APE, a developer can take network bandwidth, request frequency, data transfer requirements, server capacity, and so on into consideration. A developer can model some of the ways the application will perform and what effect different construction and deployment options present by answering questions such as , “How many client processes will connect to the server?” “How often will they make a call?” or “How long does a typical server function, such as a query, take?”

In addition, the APE is an example of a well-designed distributed application. Its Visual Basic source code is well commented and structured to serve as an example for component-based client/server applications. This source code can also be used as a starting point for building a custom multi-tier solution.

APE has been designed to help developers explore the distributed application scenarios outlined above in their environments and on the computers that will run the application. Building on these models, the Performance Explorer supports a developer controlling a variety of parameters to determine their overall impact on an application. While APE includes implementations for all distributed models using empty routines, APE is designed so a developer can easily specify component objects as the Service.

APE is a single application window made up of three panes. A developer uses the:

- **Upper pane** to load or save a profile of the connections and settings to conduct a particular test.
- **Middle pane** tabs for controlling the parameters of each of the components.
- **Lower pane** to review summary statistics.

It is a flexible and powerful tool a developer can use to get answers to interesting questions. Most importantly, a developer can directly examine the effects on throughput by varying the major input parameters, such as the number of calls, the size of objects, whether the service is in-process or out-of-process, and so on.

Other questions APE can help a developer explore include:

- What performance advantages does early binding offer over late binding?
- What will happen to the model when network capacity is exceed?
- What is the effect on performance of using asynchronous notification?

- What is the effect on performance of detailed event logging?
- What is the optimum component size for this environment? By varying the service parameters and substituting custom service components, developers can examine different component sizes and help determine how granular components should be.
- What is the best network protocol? Using the Protocol parameter available on the Connection tab, it is possible to communicate with remote objects using any installed RPC transport protocol.
- What is the appropriate level of call security? Using the Authentication parameter available on the Connection tab, it is possible to set RPC authentication to any supported value. In general, higher levels of authentication mean increased overhead. Controlling this parameter helps a developer determine the effect that call security has on an application.
- What are the queue dynamics when work is also being done on the queue system? Since developers can control whether a service uses processor cycles, they can simulate whether the queue operates on a dedicated system or is passed off to yet another remote server.
- How will the response time of clients be affected when any of these events occur?

When considering how best to take advantage of the potential for distributed computing, the first and most important question a designer must answer is whether the benefits balance the overhead associated with this approach. The response to this question can sometimes determine whether a specific application task should in fact be executed remotely.

To answer this question, a number of performance, maintenance, and administrative factors must be considered. No tool can answer all of these questions, APE can go a long way toward answering the most important performance questions.

Test Drive Visual Basic Version 5.0

Covering all of Visual Basic version 5.0 in a single test drive is impossible, so we have separated the test drive into four separate sections. The development environment and new features will be revealed while going through the individual test drives.

- **Create an ActiveX Control**—Create an ActiveX Control and use it in a Visual Basic application and an HTML page.
- **Create an Active Document**—Migrate a Visual Basic application to a Web-based, Active Document and run it in the Microsoft Internet Explorer.
- **Create an ActiveX Server Component**—Create an ActiveX Server Component and use it in a three-tier Visual Basic application and an Active Server Page.
- **Compile an ActiveX Server Component with the Native Code Compiler**—Explore the various optimizing features of the Visual Basic 5.0 native code compiler.

Note In these sections, referenced figures generally follow the text that references them.

Create an ActiveX Control

This section of the test drive is designed to give you a speedy overview of the simple process involved in creating ActiveX Controls with Visual Basic. If you are familiar with any previous version of Visual Basic you are only about ten minutes away from creating your first ActiveX Control.

When you complete these steps, you will have built what is commonly called a spinner control, seen in Figure 29. A spinner control is a graphical ActiveX Control end users employ to increment or decrement a value using a mouse instead of the keyboard.

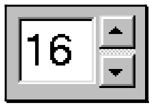


Figure 29. An ActiveX Control, called a spinner control.

Software requirements

In order to complete this section of the test drive, install:

- Microsoft Windows 95
- Any edition of Microsoft Visual Basic version 5.0; Control Creation, Professional, or Enterprise.

Setup

- Install Microsoft Windows 95 and Visual Basic version 5.0; Control Creation, Professional, or Enterprise edition.

Create a test container

In this step, you will creates the host application. This host will be used as the test container for the spinner control you will build.

1. Start Visual Basic, and in the **New Project** dialog box, choose Standard EXE, and then click **Open.**

You can see the **New Project** dialog box in Figure 30.

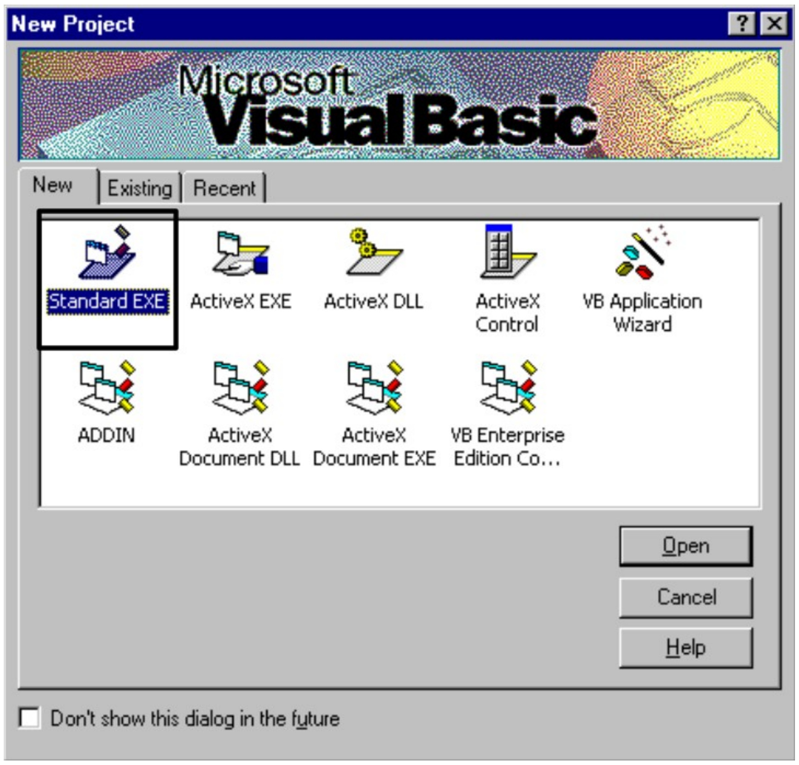


Figure 30. The New Project dialog box showing Standard. EXE highlighted.

Add a blank ActiveX Control project

1. From **File,** choose **Add Project.**
2. In the **Add Project** dialog box, choose ActiveX Control, and then click **Open.**

You can see the **Add Project** dialog box in Figure 31.

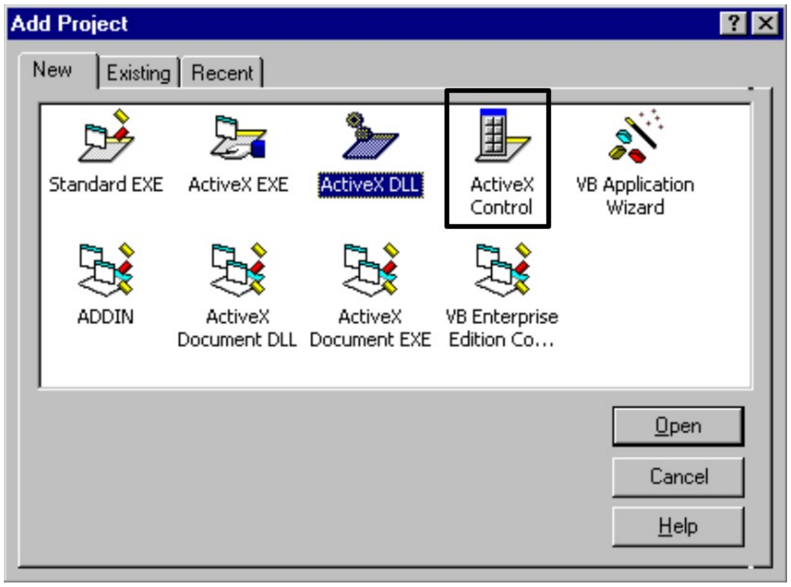


Figure 31. The Add Project dialog box, with ActiveX Control highlighted.

> **Note** Now you should have two similar looking projects open; your screen should look like Figure 32.

DataTip You can see a new control in the Toolbox, see 5 in Figure 32, named Multiple Projects. Position your mouse over this control in the Toolbox to display the DataTip giving the current name of the control, UserControl1. The icon in the Toolbox is gray or disabled now, but will become visible and enabled after a few more steps.

Visual Basic 5.0 uses the same visual metaphor for building ActiveX Controls as it uses for building applications. Using the metaphor, first you draw the interface, set properties, write event driven code, and then you can test and use the control.

Draw the visual interface for the control

Now you can create the visual interface for the spinner control. The spinner control can be created using a powerful new feature of the Control Creation Edition—the ability to combine existing controls into new, more specialized controls. To create the spinner control for our test drive, you will combine a standard textbox and a vertical scrollbar.

1. In the Visual Basic Toolbox, click the textbox control.
2. Draw a small textbox, identified as 1 in Figure 32, in the upper left corner of the Project2 window using the click-and-drag method.
3. In the Toolbox, click the vertical scrollbar control, identified as 2 in Figure 32, and draw it to the right of the textbox control—again, using the click-and-drag method.
4. Drag the control sizing handle, identified as 3 in Figure 32, to surround both newly drawn controls.

Now have a visual interface for a spinner control.

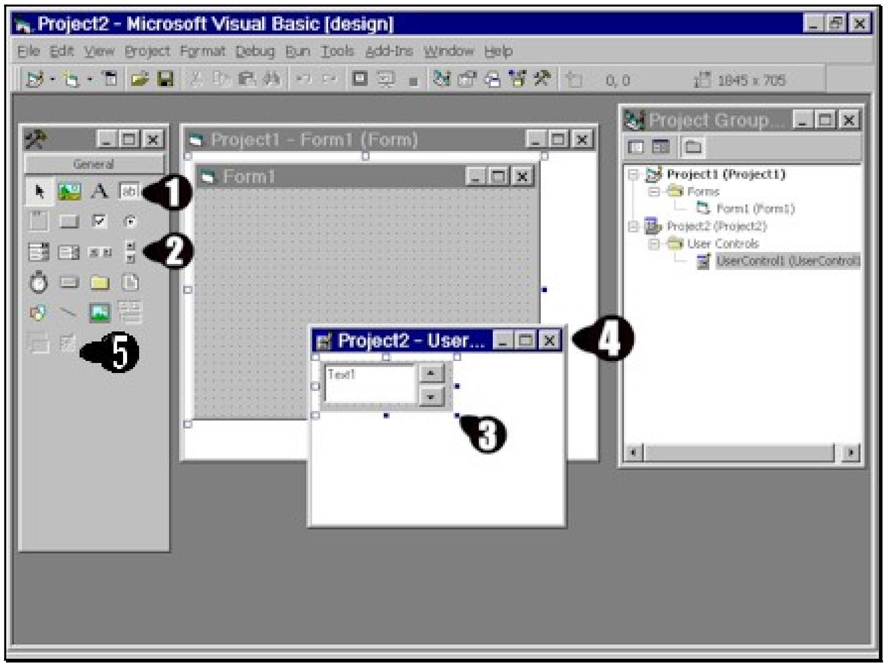


Figure 32. Project2 in Microsoft Visual Basic, showing five numbered elements in the window.

Write event-driven code

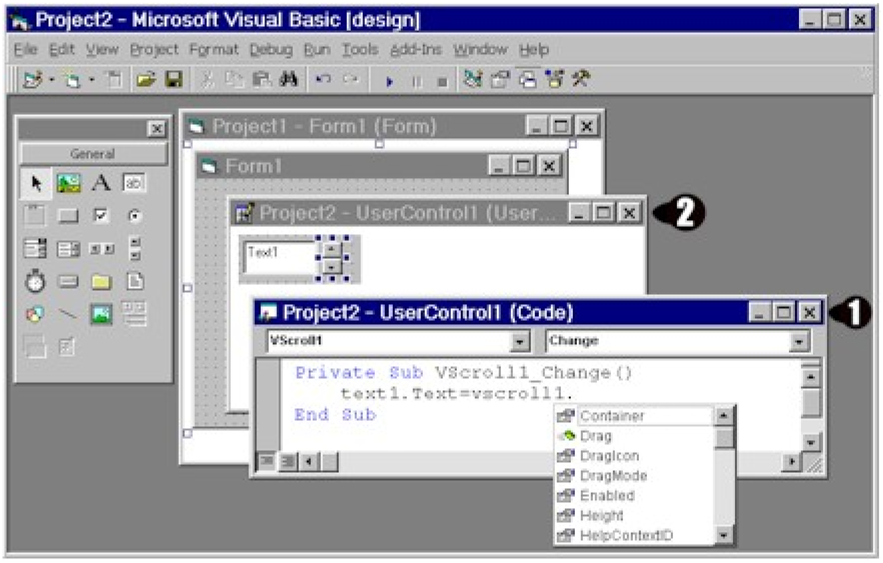
The next step is to write event driven code that will place the current value of the vertical scrollbar into the textbox. When a user clicks the up or down arrow of the vertical scrollbar, the value displayed in the textbox increments or decrements. To make this happen, some code needs to be written in the Change Event procedure for the vertical scrollbar.

1. Double click the vertical scrollbar, to display the code window. In the code window, type :

```
text1.text = vscroll1.value
```

If you were surprised as you began to type the line of code, you experienced a glimpse of the new intelligence added to the Visual Basic 5.0 development environment. As soon as you typed `text1.`, you saw a list of all allowable properties for a textbox. All ActiveX Components contain this type of information and Visual Basic makes it automatically available when needed.

2. Click the Close Box to close the Project2 code window, identified as 1 in Figure 33.
3. Click the Close Box to close the spinner control form, identified as 2 in Figure 33.


Project2 - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
General
Project1 - Form1 (Form)
Form1
Project2 - UserControl1 (User...
Text1
2
Project2 - UserControl1 (Code)
1
VScroll1
Change
Container
Drag
DragIcon
DragMode
Enabled
Height
HelpContextID

```
Private Sub VScroll1_Change()
    text1.Text=vscroll1.
End Sub
```

Figure 33. Microsoft Visual Basic, showing two numbered Close Boxes.

Use and test the control

Once you closed the spinner form, the spinner control no longer appeared gray in the Toolbox, identified as 1 in Figure 34, and is ready to be tested and used. Your development environment should look similar to Figure 34.

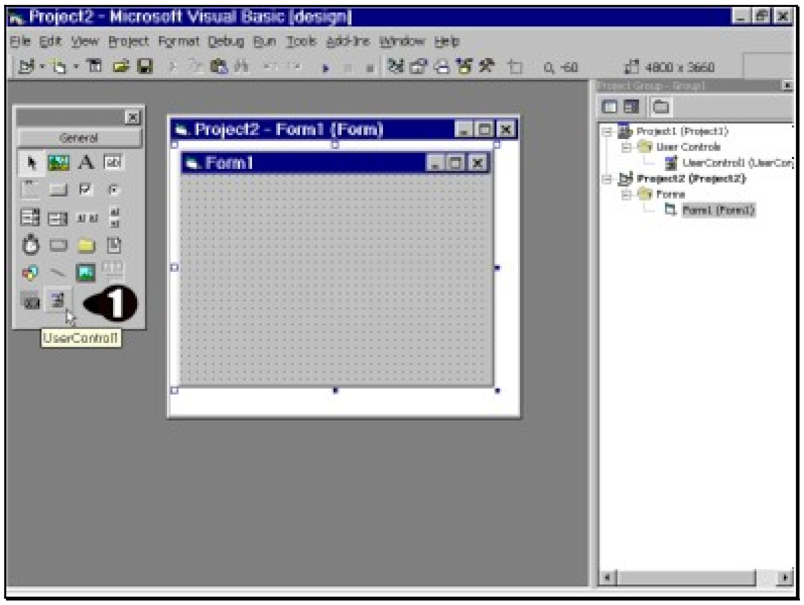


Figure 34. Visual Basic Development environment, showing the newly built spinner control as enabled.

4. Click the new user control's icon in the Toolbox and draw it on Form1.

5. Press F5 to run the application.

Note As you click the up and down arrow in the spinner control, the value in the textbox changes, just as we coded it.

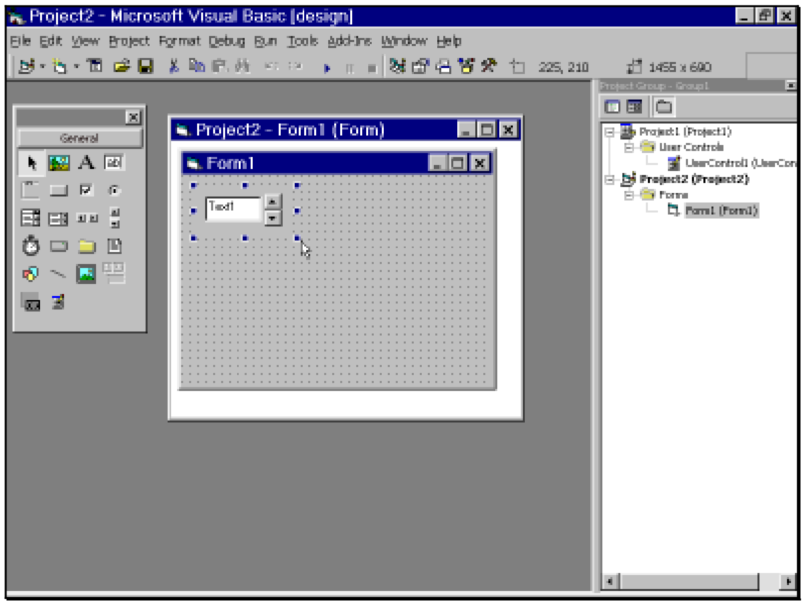


Figure 35. Visual Basic development environment, shown testing the newly built spinner control.

Congratulations, you have just created your first ActiveX Control and you only wrote one line of code.

Create an Active Document

This section of the test drive is designed to give you a speedy overview of the simple process involved in migrating a standard Visual Basic application to a Web-based, Active Document.

When you reach the end in approximately ten minutes, you will have migrated a standard Visual Basic application into an Active Document. Active Documents run in a Web browser and automatically download themselves when a user types the appropriate URL or when a link to that URL is chosen.

Software requirements

In order to complete this test drive, install:

- Microsoft Windows 95.
- Professional or Enterprise edition of Microsoft Visual Basic version 5.0.
- Microsoft Internet Explorer 3.01 or higher.
- WinZip or PKUNZIP.

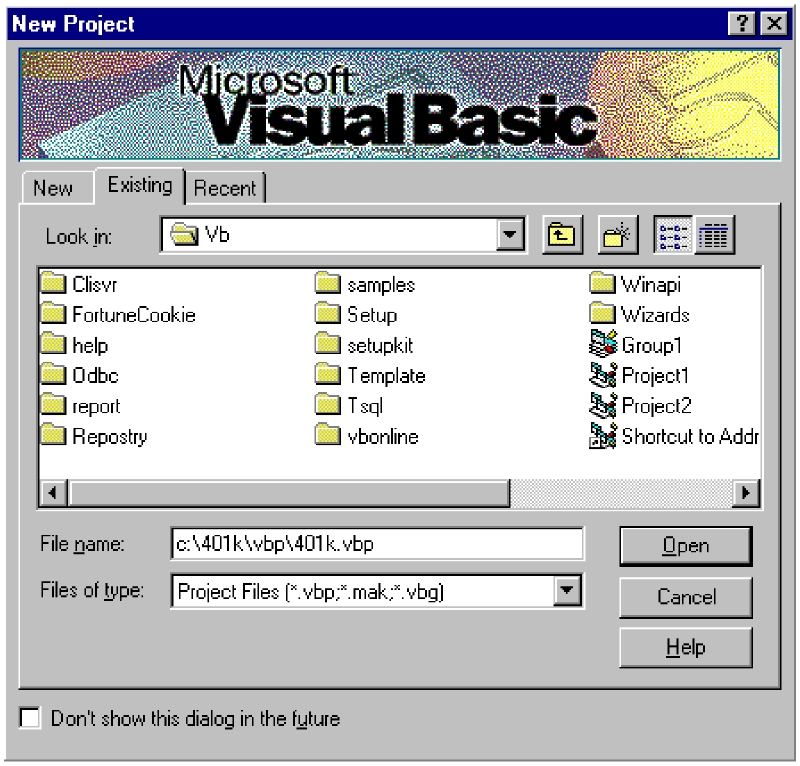
Setup

1. Install Microsoft Windows 95, the Professional or Enterprise edition of Microsoft Visual Basic version 5.0, and Microsoft Internet Explorer 3.01 or later.
2. At the root level of your hard disk, create a directory or folder, and name it 401k.
3. Copy the `401k.zip` file to the 401k directory you created.
4. Use PKZip or WinZip to extract all files from `401k.zip`.

Note You must set the extract path to `C:\` and extract options to **Use Folder Names**.

Open and run the 401K Visual Basic project

1. Start Visual Basic, and from the Template Gallery, choose **Existing.**
2. Navigate to your 401k directory, load `401k.vbp,` and then choose **Open.**

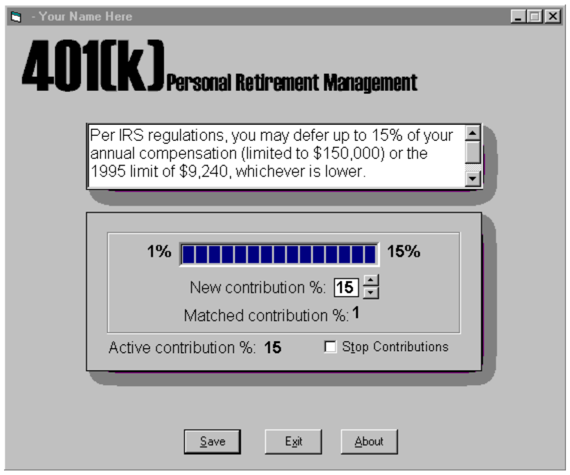


File name: c:\401k\vbp\401k.vbp

Files of type: Project Files (*.vbp;*.mak;*.vbg)

Figure 36. The New Project dialog box showing the path to the 401k.vbp file in the File name box.

3. Press F5 to run the application. You will see a form on your screen that looks like Figure 37.



Per IRS regulations, you may defer up to 15% of your annual compensation (limited to $150,000) or the 1995 limit of $9,240, whichever is lower.

New contribution %: 15

Matched contribution %: 1

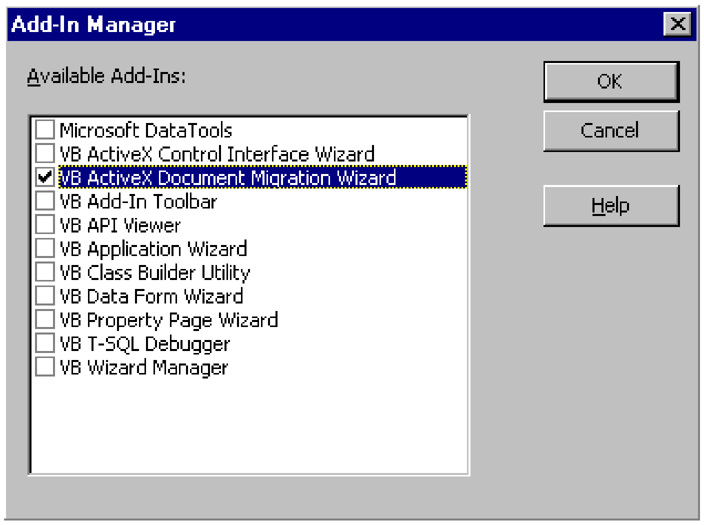
Active contribution %: 15

Figure 37. The 401(K) Personal Retirement Management enrollment form.

Now the application is running like any other Visual Basic application. You can click the up and down arrows in the spinner beside the New contribution % caption to change the value of your 401K contribution, you can choose to stop contributing, and you can choose **About** to read information about the application.

Add the Active Document Migration Wizard to the Add-ins menu

1. If you have not already stopped the 401K application, stop it by choosing **Exit** on the enrollment form.
2. From **Add-Ins,** choose **Add-in Manager.**
3. In the **Add-in Manager** dialog box, if it is not already checked (✓), check Visual Basic ActiveX Document Migration Wizard.
4. Click **OK**. This will add this wizard to the Add-Ins menu.



- [ ] Microsoft DataTools
- [ ] VB ActiveX Control Interface Wizard
- [x] VB ActiveX Document Migration Wizard
- [ ] VB Add-In Toolbar
- [ ] VB API Viewer
- [ ] VB Application Wizard
- [ ] VB Class Builder Utility
- [ ] VB Data Form Wizard
- [ ] VB Property Page Wizard
- [ ] VB T-SQL Debugger
- [ ] VB Wizard Manager

Figure 38. Add-In Manager dialog box showing the ActiveX Document Migration Wizard checked.

Run the Active Document Migration Wizard

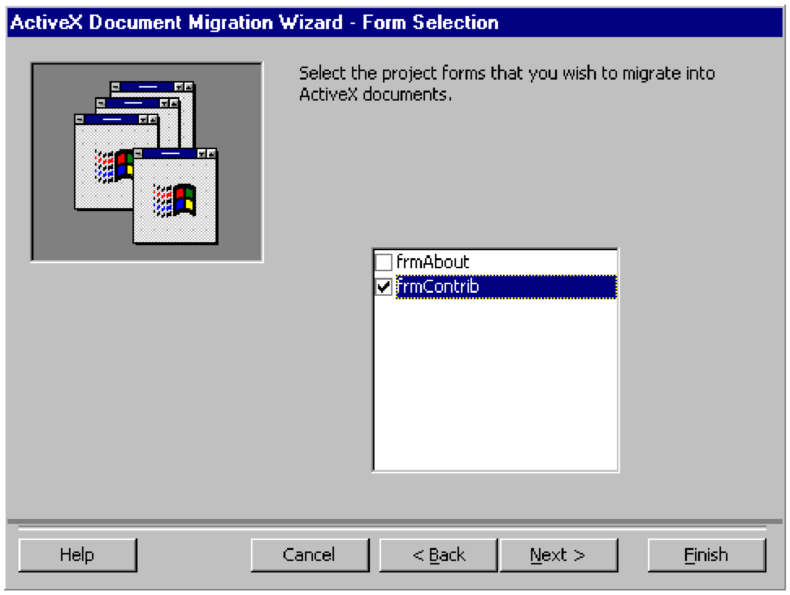
1. From **Add-Ins,** choose **ActiveX Document Migration Wizard.**

2. Click **Next** on the Introduction screen.

   Tip: This is the first screen that appears when you select the ActiveX Document Migration Wizard.

3. Check **frmContrib,** and if **frmAbout** is checked, uncheck it, and then click **Next.**

   Note This tells the wizard that you want this form to display in the browser.


ActiveX Document Migration Wizard - Form Selection

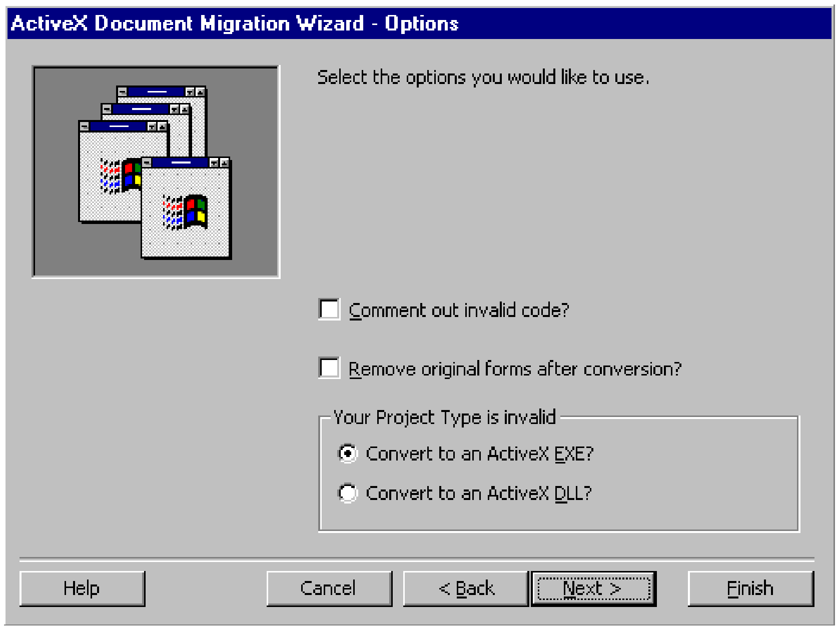
Select the project forms that you wish to migrate into ActiveX documents.

- [ ] frmAbout
- [✓] frmContrib

Figure 39. Active Document Migration Wizard Form Selection dialog box.

4. If Comment out invalid code or Remove original forms after conversion are checked, uncheck them.

5. In the Your Project Type is invalid area, choose Convert to an ActiveX EXE, and then choose **Finish**.


ActiveX Document Migration Wizard - Options

Select the options you would like to use.

- [ ] Comment out invalid code?
- [ ] Remove original forms after conversion?

Your Project Type is invalid

[✓ Convert to an ActiveX EXE?] Convert to an ActiveX DLL?

Figure 40. ActiveX Document Migration Wizard Options dialog box.

6. Click **OK** on the Active Document(s) Created Window.
7. Click **Close** on the Active Document Migration Wizard Summary Report.

   Note: Click **OK** in response to the informational message about commented-out code.

Run the 401K Active Document in the browser

Now you can run the Active Document version of the application using a Web browser. However first, we need to run the application in Visual Basic.

- Press F5 to run the application.

The application will run but a form will not display in Visual Basic because it is waiting to be hosted by the Microsoft Internet Explorer.

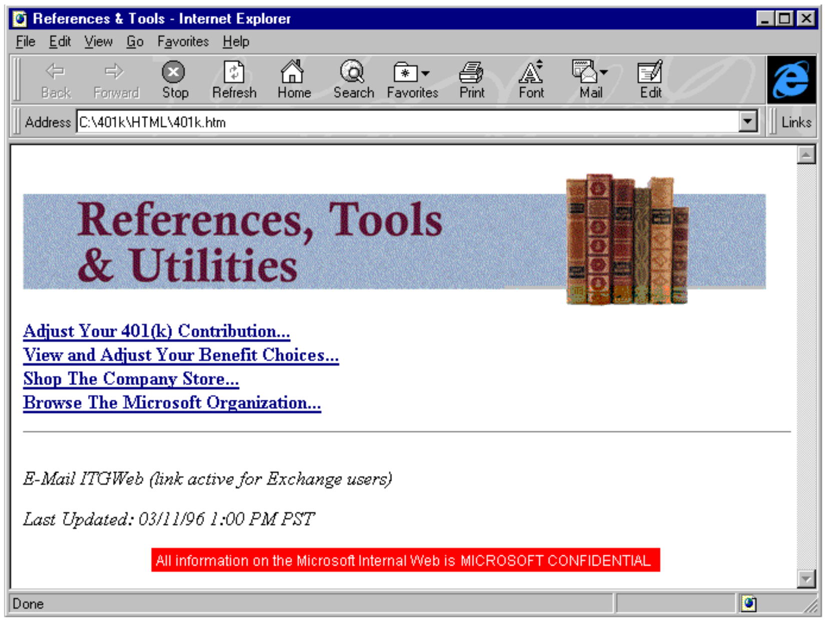
1. Start Microsoft Internet Explorer.
2. In the Address box, type:

```
C:\401k\html\401k.htm
```

   Note: If you unzipped 401k.zip to a directory other than `C:\401k`, enter that path in the Address box.

3. Click the Adjust your 401(K) Contribution hyperlink.

You can see the 401(K) Active Document running in the browser in Figure 41.


References & Tools - Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Mail Edit
Address C:\401k\HTML\401k.htm
Links
References, Tools & Utilities
Done

Adjust Your 401(k) Contribution...
View and Adjust Your Benefit Choices...
Shop The Company Store...
Browse The Microsoft Organization...

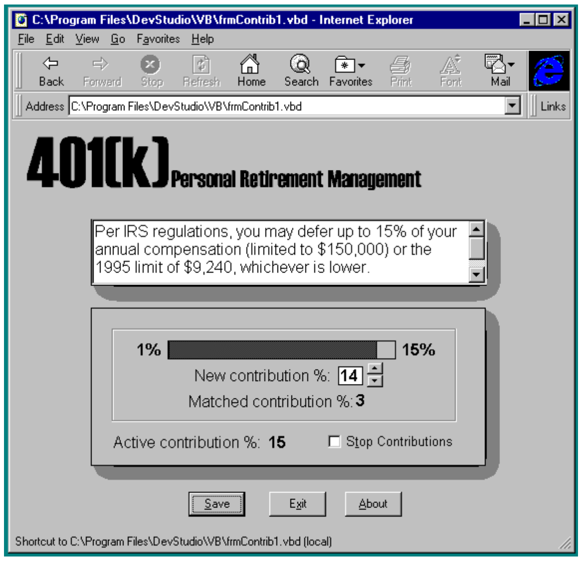
E-Mail ITGWeb (link active for Exchange users)

Last Updated: 03/11/96 1:00 PM PST

All information on the Microsoft Internal Web is MICROSOFT CONFIDENTIAL

Figure 41. The 401(K) Active Document running in the browser.

Congratulations, you have just created your first Active Document, and you didn't write a single line of code!


C:\Program Files\DevStudio\VB\frmContrib1.vbd - Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites Print Font Mail
Address C:\Program Files\DevStudio\VB\frmContrib1.vbd
Links
401(k) Personal Retirement Management
Save
Exit
About
Shortcut to C:\Program Files\DevStudio\VB\frmContrib1.vbd (local)

Per IRS regulations, you may defer up to 15% of your annual compensation (limited to $150,000) or the 1995 limit of $9,240, whichever is lower.

1% 15%
New contribution %: 14
Matched contribution %: 3
Active contribution %: 15
Stop Contributions

Figure 42. The 401(K) Visual Basic enrollment form running in the browser.

The Visual Basic application is now running in the Microsoft Internet Explorer as though it were a Web page. If the application were not already on your machine, it together with its dependent components would have been downloaded and installed on your computer when you chose the hyperlink. Additionally, if you had an old version, it would be updated with the most current version.

To learn more about how to use the Internet installation capabilities of Active Documents, read about the new Internet setup features of the Visual Basic Setup Wizard.

Close the 401K Active Document

1. Close Microsoft Internet Explorer.
2. Choose **Stop** on the Visual Basic toolbar.

   Tip: If you want to perform this demonstration again, do not save changes.

Create an ActiveX Server Component

ActiveX Server Components are Visual Basic DLLs that conform to ActiveX technologies. Although significantly enhanced in Visual Basic 5.0, the ability to create ActiveX Server Components was first introduced in Visual Basic 4.0. Recent Internet technological advancements such as Active Server Pages, however, offer a way for programmers who use Visual Basic to use their existing skills and code to enhance their Web servers with dynamic Web pages.

This section of the test drive is designed to give you a speedy overview of the process involved in creating a multi-purpose ActiveX Server Component that can be used in traditional client/server, Internet, and Microsoft Transaction Server architectures.

In this test drive, which will take approximately ten minutes, you will create an ActiveX Server Component that simply returns your fortune when called. While this server has limited practical value, it outlines the procedure for creating multi-purpose server applications.

Software requirements

In order to complete this test drive, install:

- Microsoft Windows 95.
- Professional or Enterprise edition of Microsoft Visual Basic version 5.0.

Setup

1. Install the Professional or Enterprise edition of Microsoft Visual Basic version 5.0.

2. Copy the `code1.txt` file to your desktop. This file contains approximately 20 lines of source code that you can paste into your project.

Create a test form

In this step, you will create the host application. This host will be used to test the ActiveX Server Component that you will create.

1. Start Visual Basic, and in the **New Project** dialog box, highlight Standard EXE, and click **Open**.

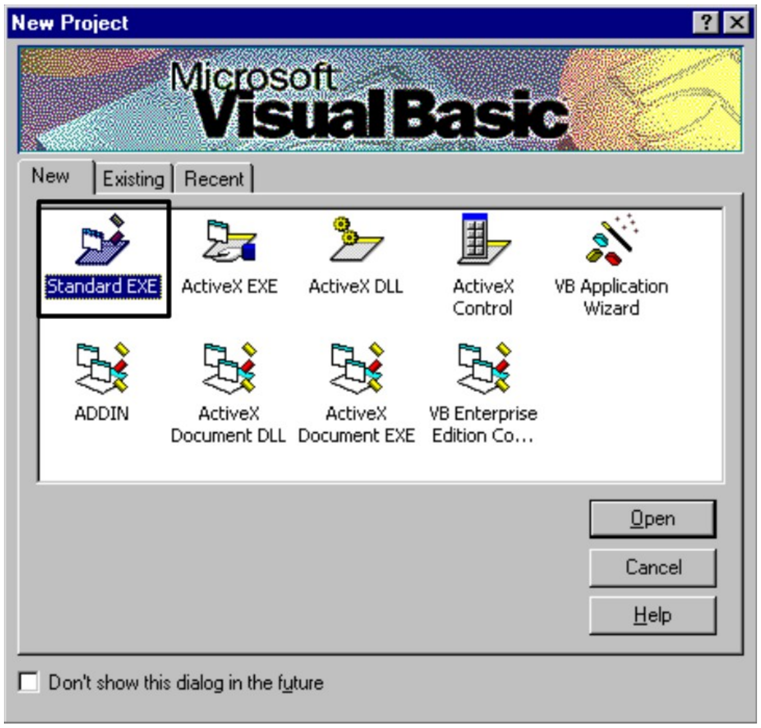


Figure 43. The New Project dialog box showing Standard EXE highlighted.

Add a blank ActiveX Server Component project

1. From **File**, choose **Add Project**.

2. In the **Add Project** dialog box, highlight ActiveX DLL, and then click **Open**.

You can see the **Add Project** dialog box in Figure 44.

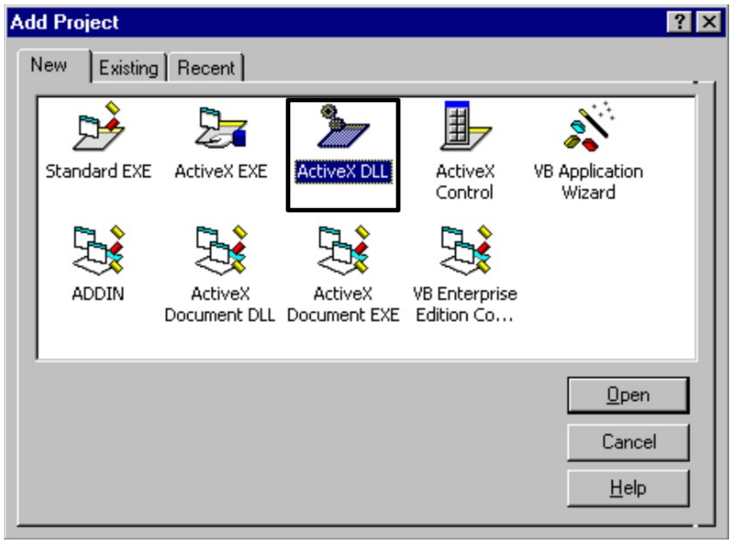


Figure 44. The Add Project dialog box, showing ActiveX DLL highlighted.

Now you have two projects open, and your screen should look like Figure 45.

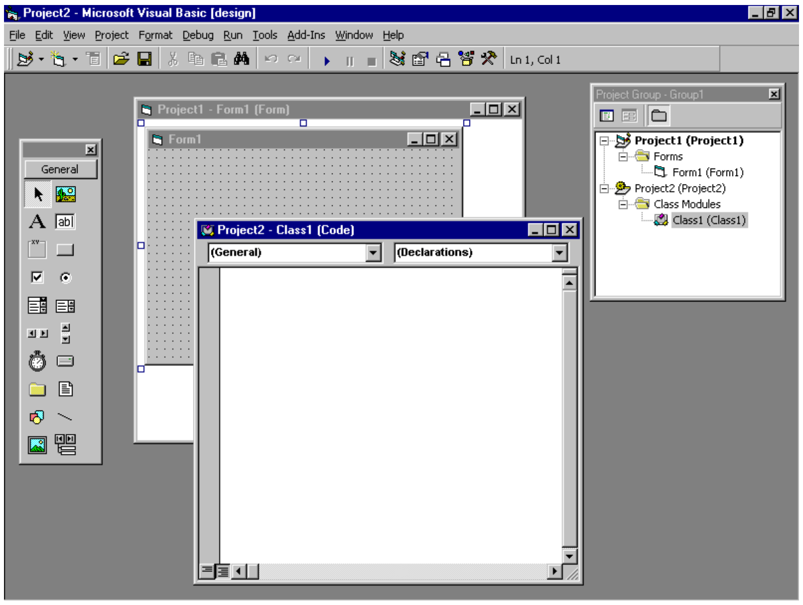


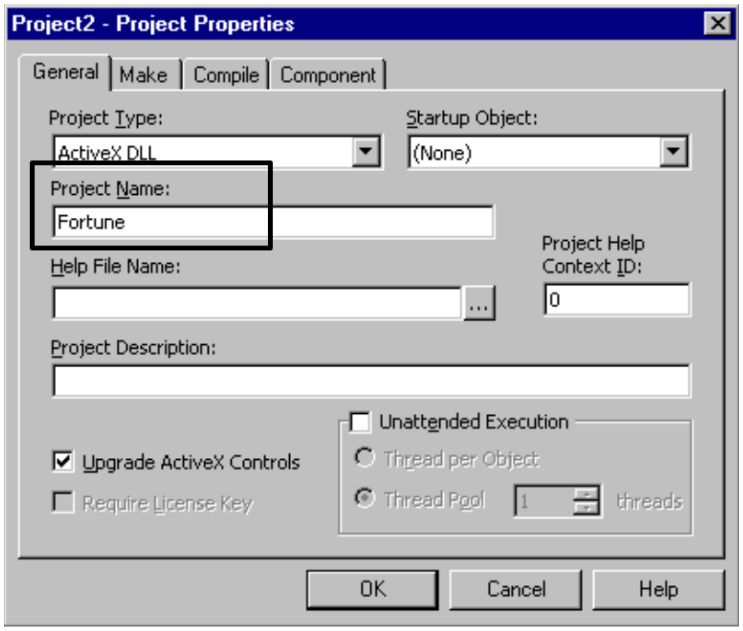
Figure 45. The Visual Basic development environment with two projects open.

Name the ActiveX Server Component and Class Module

3. Click Project2 in the Project Explorer Window using the right mouse button, and select **Project2 Properties** from the shortcut menu.

4. In the Project 2 Project Properties dialog box, in the Project Name box, change the project name to `Fortune`, and then choose **OK.**

You can see the Project2 Project Properties dialog box in Figure 46.



Project2 - Project Properties

General | Make | Compile | Component

Project Type: ActiveX DLL

Startup Object: (None)

Project Name: Fortune

Help File Name:

Project Help Context ID: 0

Project Description:

☐ Unattended Execution

○ Thread per Object

● Thread Pool 1 threads

☑ Upgrade ActiveX Controls

☐ Require License Key

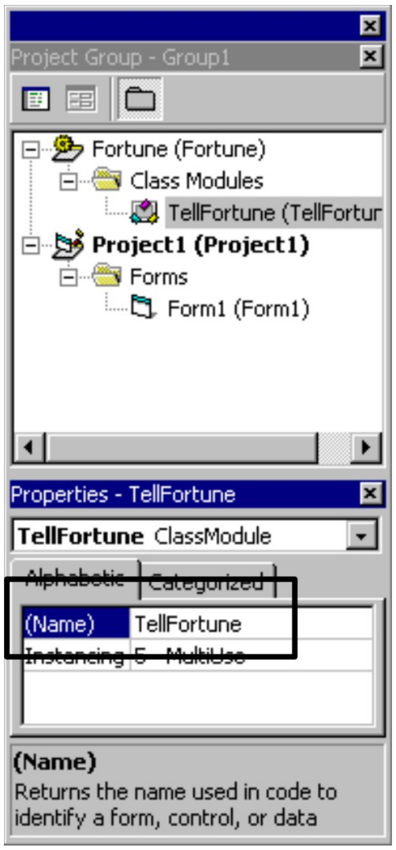
OK | Cancel | Help

Figure 46. The Project2—Project Properties dialog box.

5. Right click **Class1** under Fortune in the Project Explorer window, and choose **Properties** from the shortcut menu.

6. Change the Class Name in the Properties Window from Class1 to **TellFortune**

You can see more details in Figure 47.



(Name)	TellFortune
Instancing	5 - MultiUse

(Name)
Returns the name used in code to identify a form, control, or data

Figure 47. The TellFortune project in the Project Explorer Window and in the Properties dialog box.

Write the code to generate the Fortune

1. Double-click the **TellFortune** class module in Project Explorer Window, to open the Code window.

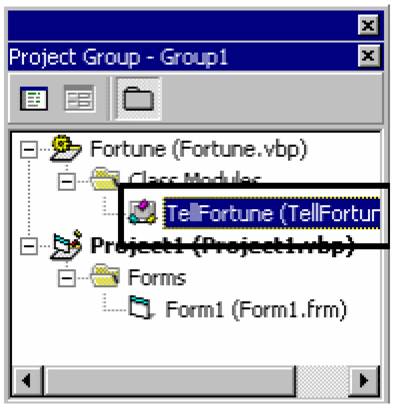


Figure 48. TellFortune highlighted in the Project Explorer Window.

2. Type this code into the Code window or paste it from the `code.txt` file.

This will create the interface that provides the lucky number and the fortune to other components.

This property provides the lucky number.

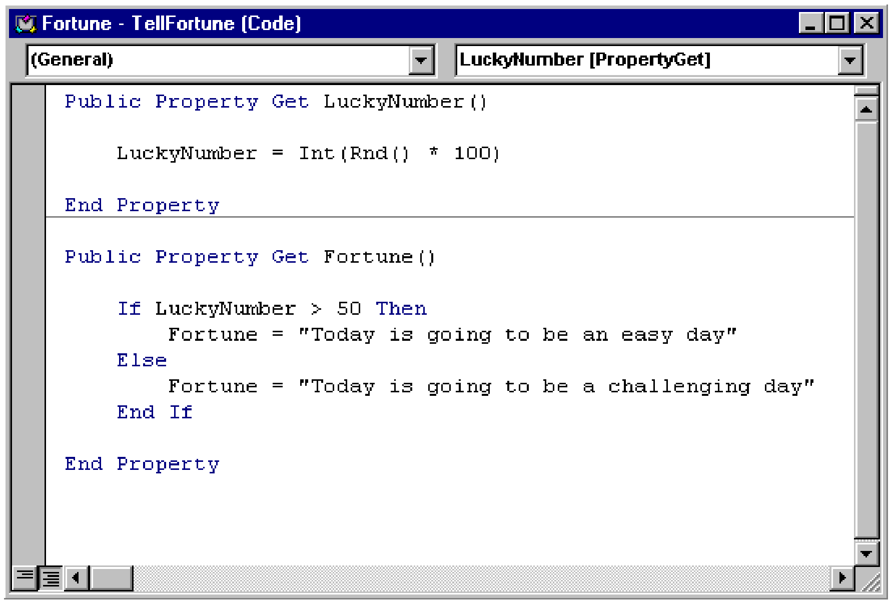
```
Public Property Get LuckyNumber()

   LuckNumber = Rnd() * 100

End Property
```

This property provides the fortune.

```
Public Property Get Fortune()

   If LuckyNumber > 50 then

      Fortune = "Today is going to be an easy day"

   Else

      Fortune = "Today is going to be a challenging day"

   End If

End Property
```

If the lucky number is greater than 50 the fortune will be an easy day, otherwise the fortune will be a challenging day.

You can view the code in the Code Window in Figure 49.



```
Public Property Get LuckyNumber()

    LuckyNumber = Int(Rnd() * 100)

End Property

Public Property Get Fortune()

    If LuckyNumber > 50 Then
        Fortune = "Today is going to be an easy day"
    Else
        Fortune = "Today is going to be a challenging day"
    End If

End Property
```

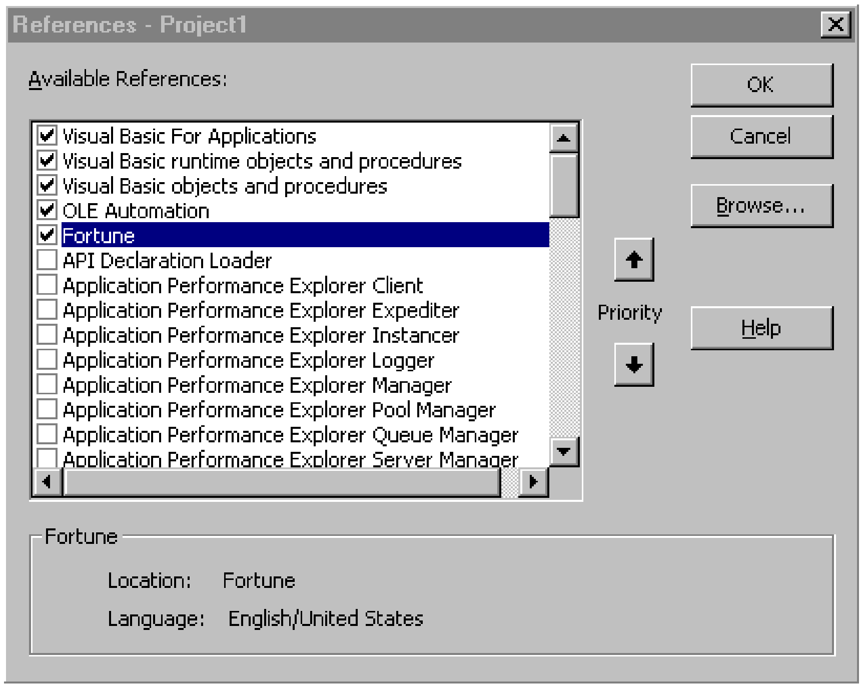
Figure 49. The Fortune TellFortune Code Window.

You have now completed the creation of the ActiveX Server. At this point, your server is ready to be invoked from an application or Web page.

Call your ActiveX Server from a Visual Basic application

In these steps you will create a form with a command button. When you click the command button, it will invoke your ActiveX Server and give you a lucky number and fortune for today.

1. Right click **Form1** in the Project Explorer Window, and choose **View Object** from the shortcut menu.
2. From **Project**, choose **References.**
3. In the **References** dialog box, check Fortune, and then click **OK.**



Available References:

- [✓] Visual Basic For Applications
- [✓] Visual Basic runtime objects and procedures
- [✓] Visual Basic objects and procedures
- [✓] OLE Automation
- [✓] Fortune
- [ ] API Declaration Loader
- [ ] Application Performance Explorer Client
- [ ] Application Performance Explorer Expediter
- [ ] Application Performance Explorer Instancer
- [ ] Application Performance Explorer Logger
- [ ] Application Performance Explorer Manager
- [ ] Application Performance Explorer Pool Manager
- [ ] Application Performance Explorer Queue Manager
- [ ] Application Performance Explorer Server Manager

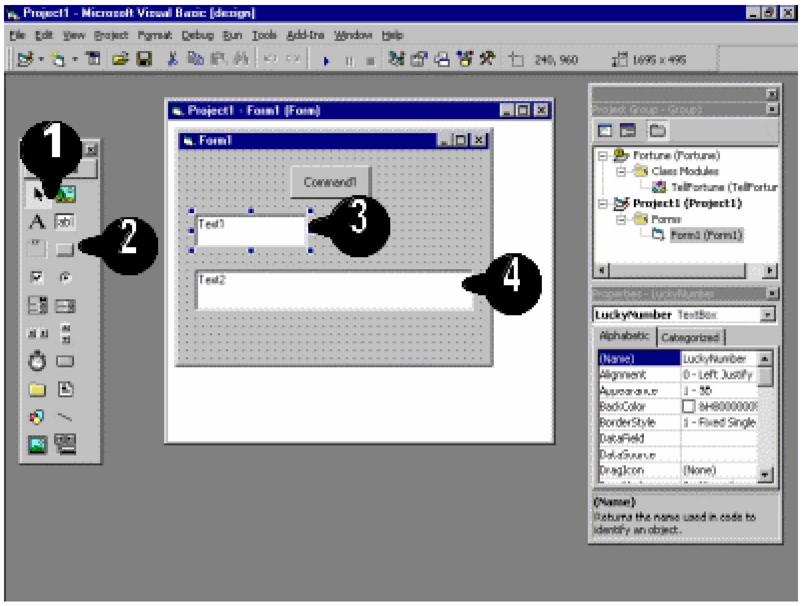
Fortune

Location: Fortune

Language: English/United States

Figure 50. The References Project1 dialog box.

4. Click the Command Button control, identified as 1 in Figure 51, in the Visual Basic Toolbox.
5. Using the mouse, draw a Command button as shown in the figure.
6. Click Textbox control, identified as 2 in Figure 51, in the Visual Basic Toolbox.
7. Using the mouse, draw a small textbox below the command button, identified as 3 in Figure 51, and then draw a larger textbox below it.
8. Click the first textbox, and in the Properties Window, change the name to LuckyNumber.
9. Click on the second textbox, identified as 4 in Figure 51, and in the Properties Window, change the name to Fortune.



(Name)	LuckyNumber
Alignment	0 - Left Justify
Appearance	1 - 3D
BackColor	&H8000000
BorderStyle	1 - Fixed Single
DataField	
DataSource	
DragIcon	(None)

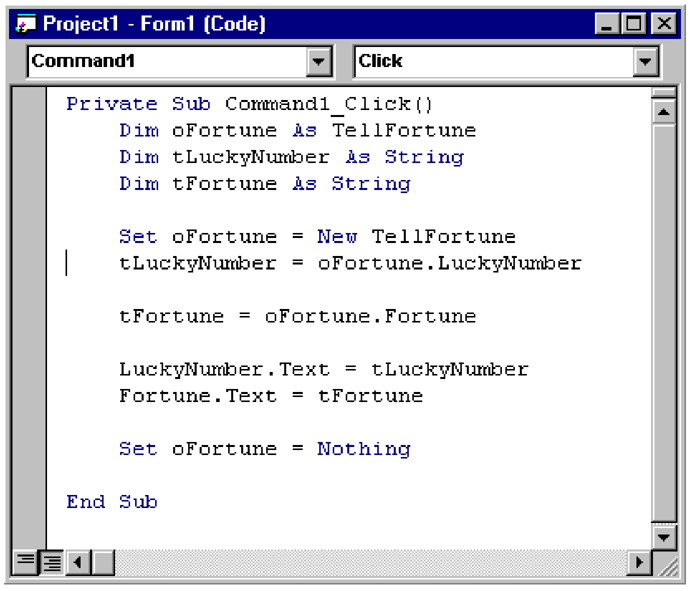
(Name)
Returns the name used in code to identify an object.

Figure 51. The Visual Basic development environment with four elements identified.

10. Double click the command button, to open the Code window for the click event.

11. Type this code or paste it from the code.txt file:

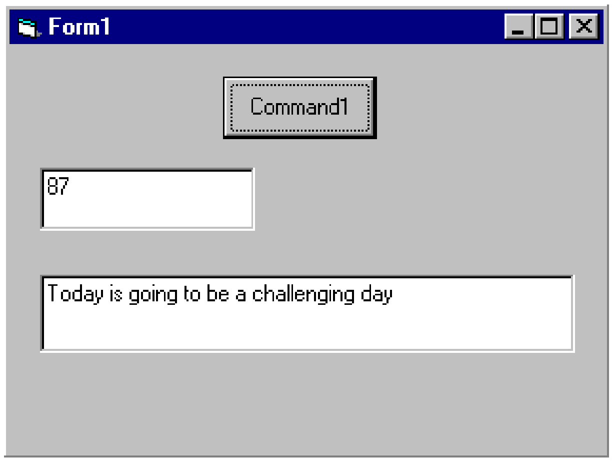
```
Dim oFortune as TellFortune

Dim tLuckyNumber as String

Dim tFortune as String

Set oFortune = New TellFortune

TLuckyNumber = oFortune.LuckyNumber


tFortune = oFortune.Fortune


LuckyNumber.Text = tLuckyNumber

Fortune.Text = tFortune


Set oFortune = Nothing
```



```
Private Sub Command1_Click()
    Dim oFortune As TellFortune
    Dim tLuckyNumber As String
    Dim tFortune As String

    Set oFortune = New TellFortune
    tLuckyNumber = oFortune.LuckyNumber

    tFortune = oFortune.Fortune

    LuckyNumber.Text = tLuckyNumber
    Fortune.Text = tFortune

    Set oFortune = Nothing

End Sub
```

Figure 52. The Project1 Form1 Code Window

Congratulations, you have now completed both the ActiveX Server Component and a Visual Basic program that uses the feature. Press F5 to run the application and click the command button several times to see the different fortunes and lucky numbers that it provides.



```
87
```

```
Today is going to be a challenging day
```

Figure 53. The Form1 ActiveX Server Component

Call your ActiveX Server from a Web page

Calling your ActiveX Server from a Web page is done with almost the exact same syntax you used above. Using three Visual Basic Scripts marked to run on the server, the following lines of code perform the same action as your form. You can see in the diagram below that your ActiveX Server Component now runs on the Internet Information Server (IIS). When Active Server Pages, an extension to IIS, recognizes that your Web page contains server-side Visual Basic script, it runs the code and dynamically builds and sends the Web page to the browser.

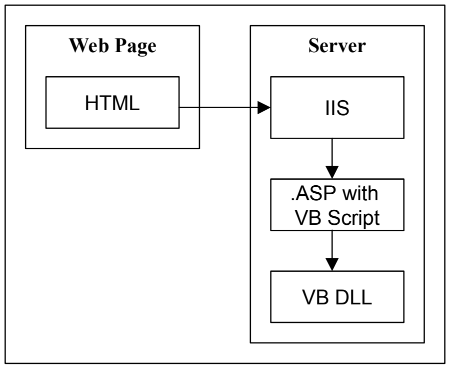


Figure 54. The Web Page-to-Server diagram.

This script, at the beginning of the Web page, invokes your ActiveX Server Component and then obtains the values.

```
Set oFortune = Server.CreateObject("Fortune.TellFortune")

      tLuckyNumber = oFortune.LuckyNumber

      tFortune = oFortune.Fortune

set oFortune = nothing
```

This script displays the Lucky Number value on the Web Page.

```
= tLuckyNumber\
```

This script displays the Fortune on the page.

```
= tFortune
```

For more information on how to embed these Visual Basic scripts in an Active Server Page look at the Web page source code included in the `code.txt` file.

Compile an ActiveX Server Component with the Native Code Compiler

This section of the test drive is designed to give you an overview of the Visual Basic Native Code Compiler.

You can explore the compiler options by compiling the ActiveX Server Component you created in the Create an ActiveX Server Component section of the test drive. If you did not complete the AcitveX Server Component section, you did not save your work from it, a copy of it has been included.

Software requirements

In order to complete this test drive, install:

- Microsoft Windows 95.
- Professional or Enterprise edition of Microsoft Visual Basic version 5.0.
- WinZip or PKUNZIP.

Setup

1. Install the Professional or Enterprise edition of Microsoft Visual Basic version 5.0.

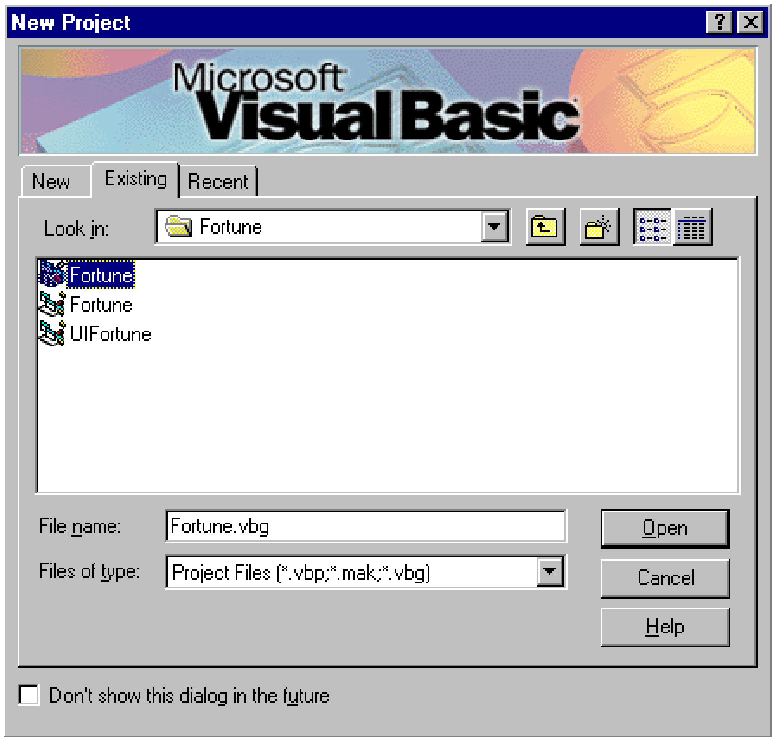
Do not perform the following steps if you plan to use the ActiveX Server Component you created in the ActiveX Server Component test drive.

2. At the root, create a directory or folder, and name it Fortune.
3. Copy the `fortune.zip` file to the Fortune directory you created.
4. Use PKZip or WinZip to extract all files from `fortune.zip`.

   Note You must set the extract path to `C:\` and extract options to **Use Folder Names**.

Open the Fortune Visual Basic project

1. Start Visual Basic and from the Template Gallery, choose **Existing.**
2. Navigate to your Fortune directory and load the first file in the list, `Fortune.vbg.`
3. Click **Open.**



Look in: Fortune

Fortune
Fortune
UIFortune

File name: Fortune.vbg

Files of type: Project Files (*.vbp;*.mak;*.vbg)

Don't show this dialog in the future

Figure 55. The New Project dialog box, with the Existing tab selected, and Fortune highlighted.

Review the various compiler options

1. Click **Fortune** in the Project Window.
2. From **Project,** choose **Fortune Properties.**

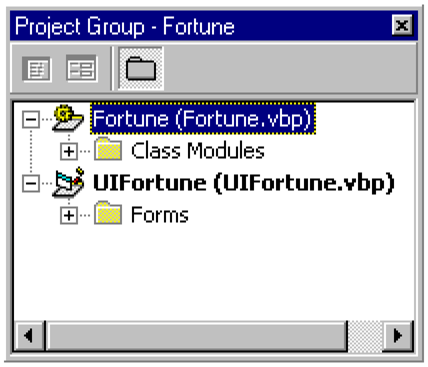
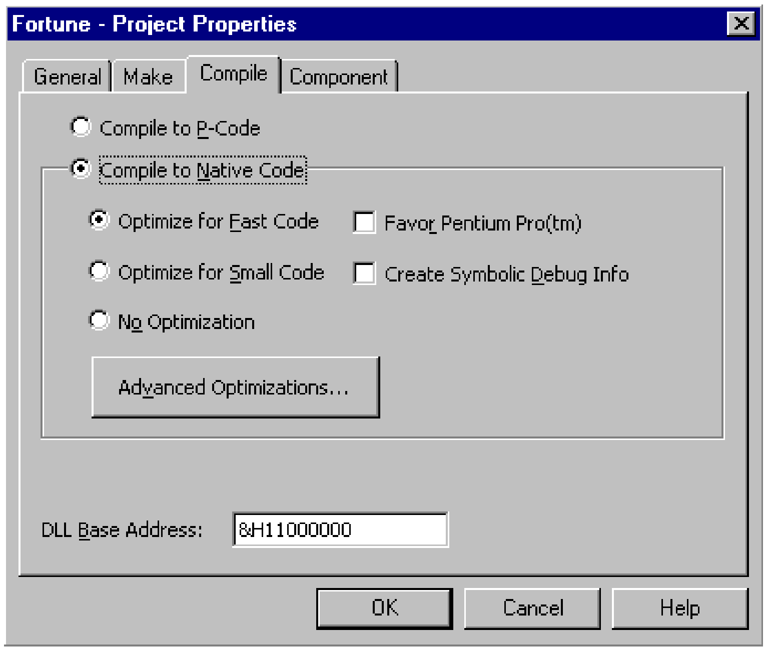


Figure 56. The Project Group Fortune window.

3. In the **Fortune Project Properties** dialog box, choose the **Compile** tab.



( ) Compile to P-Code
(●) Compile to Native Code
(●) Optimize for Fast Code
☐ Favor Pentium Pro(tm)
( ) Optimize for Small Code
☐ Create Symbolic Debug Info
( ) No Optimization
DLL Base Address: &H11000000

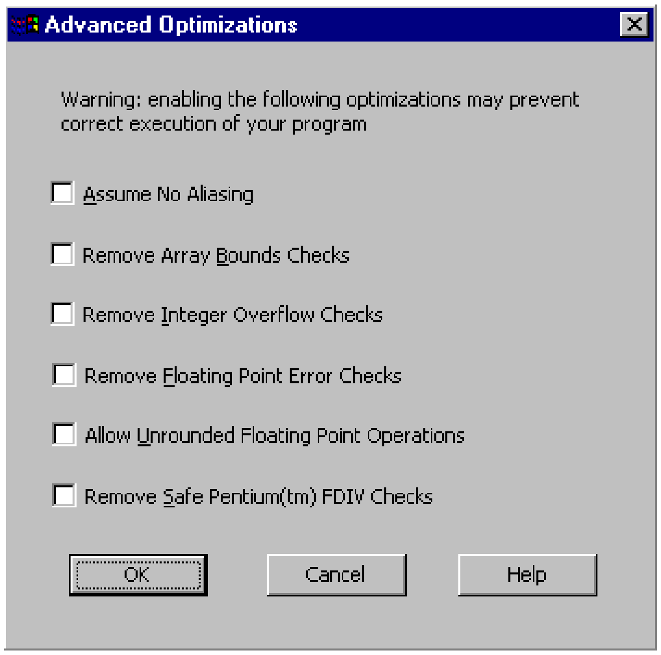
Figure 57. The fortune—Project Properties dialog box showing the Compile tab selected.

You can use this dialog box to set conditions for your project. The compiler will optimize your executable file for speed, smaller file size, or no optimization. You can also set the compiler to take advantage of the new Intel Pentium Pro features and to create symbolic debug information. Executable files created with the Create Symbolic Debug Info option checked can be debugged with Visual C++ executable files either in Developer Studio or in any other debugger that uses the CodeView style of displaying debug information

Look at advanced optimizations

1. In the **Project Properties** dialog box, click **Advanced Optimizations.**

You can review the Advance Optimizations dialog box in Figure 58.



Advanced Optimizations

Warning: enabling the following optimizations may prevent correct execution of your program

- [ ] Assume No Aliasing
- [ ] Remove Array Bounds Checks
- [ ] Remove Integer Overflow Checks
- [ ] Remove Floating Point Error Checks
- [ ] Allow Unrounded Floating Point Operations
- [ ] Remove Safe Pentium(tm) FDIV Checks

Figure 58. The Advanced Optimizations dialog box.

Visual Basic has always shielded developers from many of the more critical coding errors such as Integer overflow, floating point errors, and out-of-bounds arrays. Visual Basic does this by adding additional error routines that check and then handle these errors, so Visual Basic can trap the error. With Version 5.0, however, you have the option to create executable files that do not contain these additional routines—allowing applications to run faster.

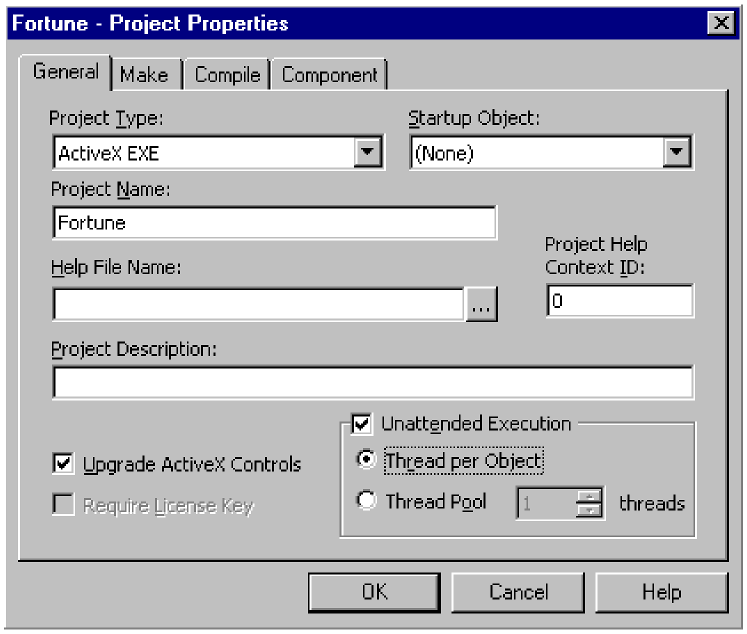
- If your application doesn't use arrays or if you are performing your own array bounds checking, check Remove Array Bounds Checking.
- If you are not using floating point variables, check Remove Floating Point Error Checks.

Since our application does not perform any of the functions that Visual Basic automatically checks for, we can check all of the advanced optimization boxes.

2. Click each of the advanced optimization check boxes, and then choose **OK.**

Look at implicit multi-threading

1. In the **Fortune Project Properties** dialog box, click the General tab.
2. Click the down arrow for Project Type, and choose **ActiveX EXE**.
3. Check Unattended Execution.


Fortune - Project Properties

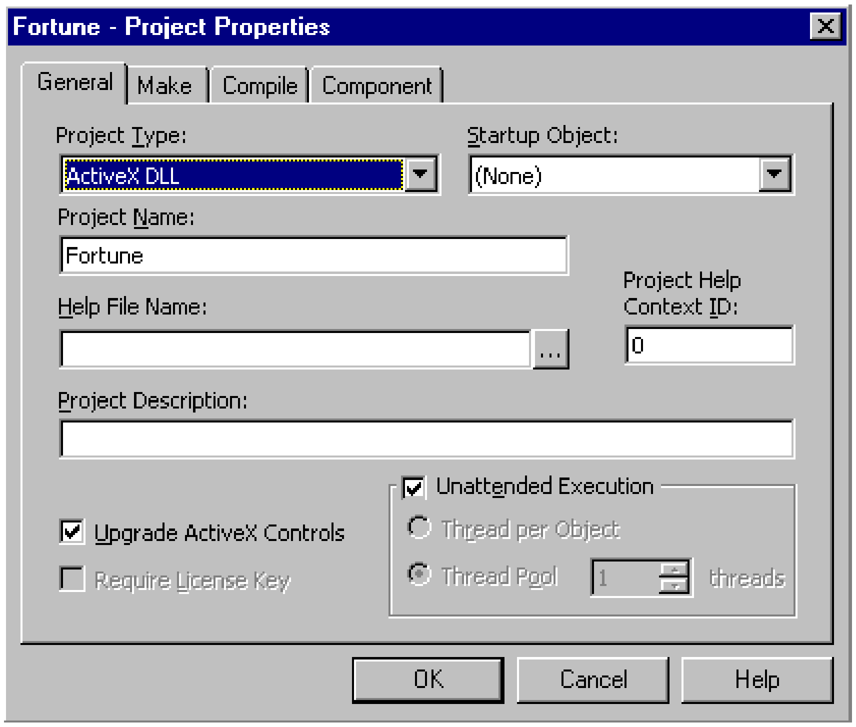
Project Type: ActiveX EXE
Startup Object: (None)
Project Name: Fortune
Help File Name:
Project Help Context ID: 0
Project Description:
[✓] Upgrade ActiveX Controls
[ ] Require License Key
[✓] Unattended Execution
● Thread per Object
○ Thread Pool 1 threads

Figure 59. The Fortune Project Properties dialog box.

Multi-threaded executable files can be created by checking the Unattended Execution box. Once set to unattended execution, you can choose to create every object in a separate thread, or to create a thread pool to be shared between multiple objects. This gives you immediate access to the multi-threading power of Windows NT.

Create a native node ActiveX Server Component

1. Click the down arrow for Project Type, and choose **ActiveX DLL**.
2. Click **OK.**
3. From **File,** choose **Make Project DLL**.
4. Click **OK.**


Fortune - Project Properties

Project Type: ActiveX DLL

Startup Object: (None)

Project Name: Fortune

Help File Name:

Project Help Context ID: 0

Project Description:

[✓] Upgrade ActiveX Controls

[ ] Require License Key

[✓] Unattended Execution

( ) Thread per Object

(●) Thread Pool 1 threads

Figure 60. The Fortune Project Properties dialog box.

Congratulations, you have just created your first Native Code, multi-threaded Visual Basic ActiveX Server Component.

Appendix A: Client/Server Development Tool Performance Analysis

Technical Report 61230-3

January 21, 1997

Carnegie Technologies, Inc.

Software Performance Engineering Division

CARNEGIE
TECHNOLOGIES

Carnegie Technologies, Inc. is a premier provider of leading edge information technology services. Although based in Atlanta, the national reputation of its principals and their commitment to excellence has attracted national attention.

At Carnegie Technologies, we employ a multi-disciplinary approach to applied information technology. Most Carnegie consultants hold doctorates from leading research universities in applied technical fields. Computer and systems science, applied mathematics, economics, and management science disciplines are blended with people-oriented fields of cognitive psychology, philosophy, and organizational behavior. The result is an organization uniquely positioned to help its clients create value through technology.

The unique structure and mission of the company has attracted innovative consultants from across the country. Counted among its consulting staff are former instructors from top universities including MIT, Duke University, Georgia Institute of Technology, Carnegie Mellon University, Emory University, and the Harvard Business School. Carnegie consultants are researcher/practitioners - many are noted speakers and authors.

Carnegie's software performance engineering division specializes in modeling performance of client/server applications. Using our unique APER™ (Application Performance Review) methodology and our extensive library of development tool benchmarks, Carnegie performance engineers are able to help our clients locate performance bottlenecks and optimize client/server applications during system design, reducing costly rework at later stages.

To receive information about other reports in this series, or to find out how Carnegie Technologies can help engineer performance into your enterprise-wide client/server applications please call:

Mr. Rob Smith, VP Marketing
770 916-0595 x405
smith@carnegie.com

Executive Summary

A new generation of client-side development tools combine the best attributes of both interpreters and compiled languages. Rapid application development is achieved through the use of an interpreted environment with the ability to instantly run and debug code as changes are made. When the application is ready for deployment, compilation provides compact code and superior application performance.

In selecting a client/server development tool, organizations are interested in the performance characteristics relative to other similar tools on the market. Carnegie has developed the Q1 performance benchmark to measure the performance of the new generation of compiled client/server development tools.

The Q1 benchmark measures application performance along four performance dimensions:

- Graphical display performance
- Database access performance
- OLE Automation performance
- Language performance
-

Each of these dimensions, in turn, is defined by a set of metrics each of which measures a specific performance characteristic of that dimension. The Carnegie Q1 standard comprises three graphical display tests, three database access tests, and four OLE Automation performance tests. In addition, the Q1 test suite contains two functions designed to emulate “real world” applications. These functions provide an overall performance benchmark of typical programming operations including numerical calculations, branching, looping, dynamic memory allocation (through variable-length strings and arrays), and string manipulation functions.

This report provides the Q1 results for the following four 32-bit development tools:

- Visual Basic 5.0
- Visual Basic 4.0 (interpreted)
- Delphi Client/Server 2.0
- PowerBuilder 5.0

All tools were run under the Windows 95 operating system.

The results show that Visual Basic 5.0 provides extremely efficient compilation of computationally intensive code as evidenced by the Sieve of Erastostenes test, which finds VB 5.0 almost ten times faster than Delphi and almost 50 times faster than PowerBuilder. Visual Basic 5.0 turned in the fastest performance in eight out of the twelve tests and a second place finish in two out of the other four. PowerBuilder was the slowest on most of the tests but did turn in a first place result on the database overhead test which measures the time required to set up a database call. Delphi turned in respectable times on most tests but was very slow at performing in process remote OLE automation calls. Delphi, however, was by a large margin, the most efficient language at string manipulations, likely due to its underlying string and memory management architecture.

VB 5.0 shows a marked improvement over VB 4.0 across the board. The most impressive improvement was in the sieve, which measures speed of control logic and numerical computation, where VB 5.0 was over 15 times faster.

Note This is an abridged version of a complete report.

Performance Benchmark Results

The results from the benchmark are summarized in the following table. The figures presented are the average time to perform a single operation, measured in milliseconds. The results for all of the categories are rounded to three decimal places.

Each test conformed to a 95% confidence interval that represented less than 1% of the average for that test. that is to say, any test could be repeated with a 95% certainty of obtaining a score within 1% of the reported value.

	Test	Visual Basic 5.0	Visual Basic 4.0	Delphi 2.0	Power Builder 5.0
Display	Empty Display	52.509	68.677	26.851	85.667
	Labels	.316	.755	1.295	4.748
	Bitmaps	1.226	2.42	3.296	8.925
Database	Overhead	9.745	22.234	7.838	6.935
	Small Retrieval	.340	1.415	.612	.393
	Large Retrieval	.251	.218	.110	.161
OLE Automation	In Process Let	.000294	.000648	.084	N/A
	In Process Method	.000295	.000482	.082	N/A
	Out of Process Let	1.066	1.439	2.572	N/A
	Out of Process Method	1.039	1.388	2.572	N/A
Language	Sieve of Eratosthenes	360.625	5622.430	3323.820	16940.120
	String Manipulation	900.350	1463.990	48.130	1941.360

Conclusions

All in all, these tools turned in excellent performance characteristics. Taken as a whole, some general conclusions can be drawn.

- VB 5.0 provides a significant speed improvement over VB 4.0, especially for numerically intensive or computationally bound applications.
- Visual Basic 5.0's numerical manipulation speed was notably faster than all competitors.
- The use of early vtable binding provides tremendous speed advantage for VB in invoking In-Process OLE servers.
- PowerBuilder performed slowest in 5 of 8 tests which it ran and was unable to perform the OLE automation calls due to a software problem which at the time of this writing remain unresolved by Powersoft's technical support staff.
- Delphi's string manipulation speed was significantly faster than VB or PowerBuilder. Note that the alternative byte array implementation in VB would have yielded improved performance—but there are no built-in string functions for manipulating them in the current implementation.

One word of caution is in order. Application performance is but one dimension of tool selection. Other important areas to be considered include:

- Vendor stability and technical support
- Market penetration and installed base
- Pool of available talent
- Ease of use
- Training costs
- Third-party support
- Tool stability and robustness
- Quality of the product's development environment (IDE)

These issues, along with code execution performance will provide a balanced set of selection criteria for client-side application development tools.

Appendix B: System Requirements

To run Visual Basic Version 5.0, Enterprise Edition, you need:

- PC with a 486/66 MHz or higher processor (Pentium or higher processor recommended) or any Alpha processor running Microsoft Windows NT Workstation
- Microsoft Windows 95 operating system, Microsoft Windows NT Workstation operating system version 4.0 (Service Pack 2 recommended), or Microsoft Windows NT Workstation operating system version 3.51 Service Pack 5*
- 16 MB of RAM for Windows 95; 32 MB for Windows NT Workstation
- Hard-disk space required:
  - Minimum installation: 35 MB
  - Maximum installation of all applications and Books Online: 345 MB
- CD-ROM drive
- VGA or higher-resolution monitor
- Microsoft mouse or compatible pointing device

To run Visual Basic Version 5.0, Professional Edition, you need:

- PC with a 486DX/66 MHz or higher processor (Pentium or higher processor recommended)
- Microsoft Windows 95 operating system, Microsoft Windows NT Workstation operating system version 4.0 (Service Pack 2 recommended), or Microsoft Windows NT Workstation operating system version 3.51 with Service Pack 5*
- 12 MB of RAM for Windows 95 or Windows NT Workstation 4.0 (16 MB recommended); 24 MB for Windows NT Workstation 3.51
- Hard-disk space required:
  - Minimum installation: 30 MB
  - Complete installation: 115 MB (220 MB required if installing online documentation)
- CD-ROM drive
- VGA or higher-resolution monitor
- Microsoft mouse or compatible pointing device

To run Visual Basic Version 5.0, Learning Edition, you need:

- PC with a 486DX/66 MHz or higher processor (Pentium or higher processor recommended)

- Microsoft Windows 95 operating system, Microsoft Windows NT Workstation operating system version 4.0 (Service Pack 2 recommended), or Microsoft Windows NT Workstation operating system version 3.51 with Service Pack 5*
- 8 MB of RAM for Windows 95 or Windows NT Workstation 4.0 (16 MB recommended); 16 MB for Windows NT Workstation 3.51
- Hard-disk space required:
  - Minimum installation: 25 MB
  - Complete installation: 80 MB (205 MB required if installing online documentation and *Learn Visual Basic Now)*
- CD-ROM drive
- VGA or higher-resolution monitor
- Microsoft mouse or compatible pointing device

**The Internet features hyperlink and AsyncRead are supported on Windows 95 and Windows NT Workstation 4.0 only.*

Appendix C: New Features Table

Table 1. New Features in Visual Basic version 5.0

Visual Basic Feature	**Benefit**				
Customizable Development Environment	Increase productivity with IntelliSense features such as Quick Info, DataTips, List Properties/Methods, and Complete Word. They eliminate the need to memorize complex syntax, optional arguments, and component properties. Plus, the new multiple-documents interface (MDI) reduces cumbersome key strokes and mouse clicks.	✓	✓	✓	✓
Visual Basic, Applications Edition, version 5.0	A developer can use Visual Basic skills in Microsoft Office 97 and third-party applications licensing Visual Basic, Applications Edition, because they use the same language and development environment that's in Visual Basic version 5.0.	✓	✓	✓	✓
Project Templates, Sample Projects, and Books Online	Take advantage of new technologies with project templates that set properties for varying project types and sample projects for reference. Books Online offers one-touch access to comprehensive documentation.	✓	✓	✓	✓
Sophisticated Add-ins	Build and manipulate class modules, properties, events, and methods. Visually define ActiveX Control interfaces, design property pages, and extrude sub-classed control members.	✓	✓	✓	✓
Multiple Projects	Develop and debug components and applications that use these components simultaneously in one instance of Visual Basic as easily as you creating monolithic applications.	✓	✓	✓	✓
Interface and Visual Inheritance	Achieve a new degree of productivity by sub-classing and aggregating ActiveX Controls. Extend existing classes, properties, events, and methods by inheriting interfaces from other class modules.	✓	✓	✓	✓
ActiveX Control Creation	Create ActiveX Controls that run in Visual Basic, Visual C++, Microsoft Office, Visual FoxPro, Web applications, and any ActiveX-aware product.	✓	✓	✓	✓
Rich Set of Standard Controls	Includes controls such as a data-bound grid, check box, combo box, file list box, drive list box, common dialog box controls, Web Browser control, WinSock and more than 20 others.	✓	✓	✓	

Visual Basic Feature	Benefit				
Application Creation	Compile and deploy standalone applications, .exes, and .dlls.	✓	✓	✓	
Learn Visual Basic Now	Jump-start the learning process with a self-paced, instructional CD-ROM that includes multimedia video lessons, narrated demonstrations, and hands-on lab exercises.			✓	
Data Access Flexibility	Enable DAO applications to achieve near-RDO performance when accessing ODBC data sources. DAO version 3.5 provides enhanced capabilities for ISAM and ODBC data sources.	✓	✓		
Optimizing Native Code Compiler	Build native code applications and components that leverage the world-class compiler technology available in Microsoft Visual C++. Applications created in version 5.0 run up to 20 times faster than in previous versions.	✓	✓		
Active Document Creation	Create browser-based applications from scratch or from existing Visual Basic applications that are hosted in Microsoft Internet Explorer and installed automatically.	✓	✓		
ActiveX Server Components	Create thread-safe and high-performance multi-threaded server components for maximum application scalability.	✓	✓		
Microsoft Transaction Server, Developer Edition	Build three-tiered component applications, including integrated-transaction and object-management systems. (For Windows NT only)	✓			
Native RISC support	Build high-performance, scalable applications and components that run natively on Alpha processor based computers.	✓			

Visual Basic Feature	Benefit				
Remote Data Objects (RDO) version 2.0	Use RDO for high-speed ODBC database access from Internet or client/server applications using an object-based programming interface.	✓			
T-SQL Debugger	Debug SQL Server stored procedures interactively, at the same time, and on the same workstation, where developing and debugging Visual Basic applications occurs.	✓			
Microsoft SQL Server, Developer Edition	Build and test prototype SQL Server 6.5 applications on a LAN (for Windows NT Workstation).	✓			
Microsoft Visual SourceSafe	Manage source code using an integrated code management system; including check-in/check-out, visually view differences, resource sharing, and project histories for team development.	✓			
Application Performance Explorer (APE)	Simulate and analyze distributed application scenarios before committing to specific architectures and implementations, for optimum performance.	✓			
Microsoft Repository	Extend the development environment or develop custom repository applications.	✓	✓		
Microsoft Visual Database Tools	Design and build databases—all with one integrated tool. Also use Database Tools to build and fine-tune client-side queries and server-side data structures including tables, relations, and stored procedures.	✓			